

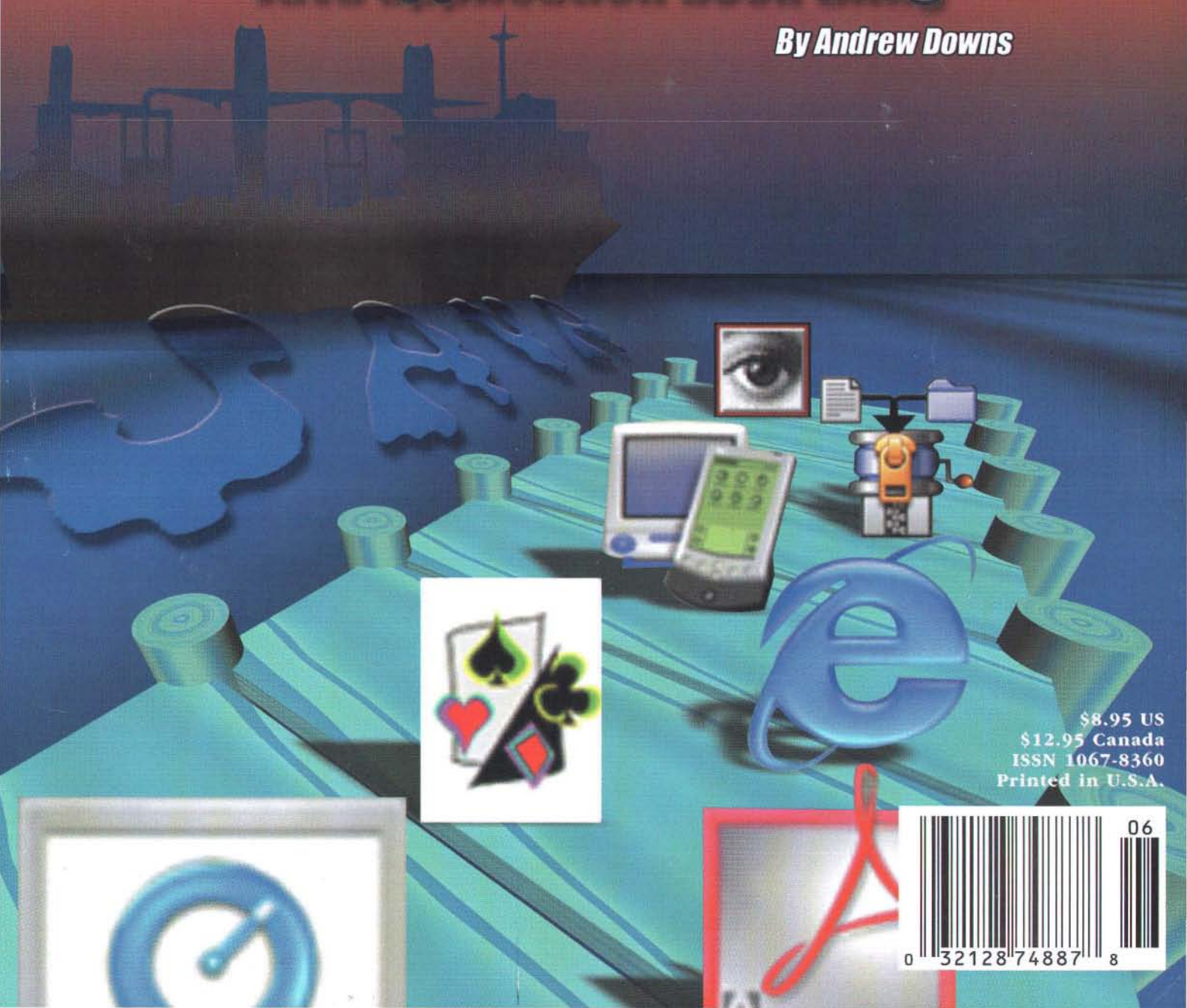
MacTech Magazine
Vol. 18, No. 06 • June 2002

MacTech®

The Journal of Macintosh Technology and Development

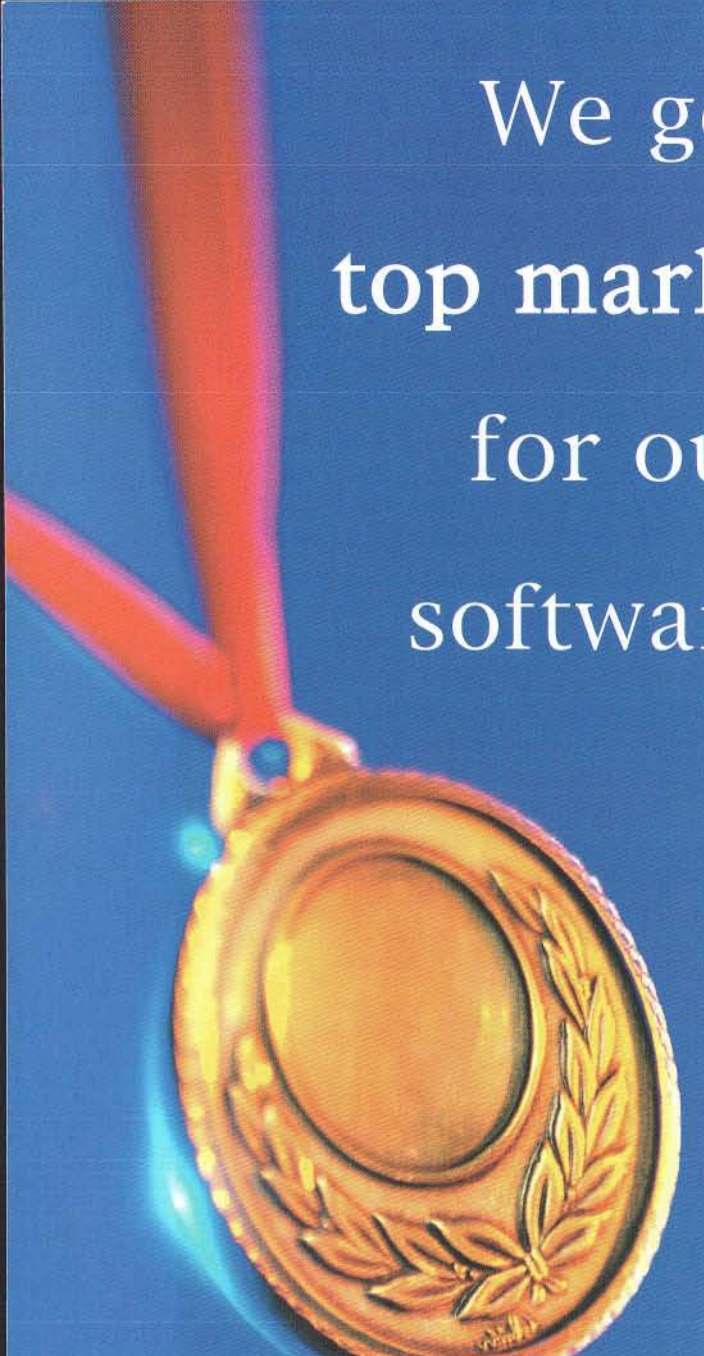
Change your image! Java application dock tiling

By Andrew Downs



\$8.95 US
\$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.





We got
top marks
for our
software.

You can too.

Revolution™
The Solution for Software Development
In enterprise, business and education

REVOLUTION™

Limitless possibilities

With Revolution, the solutions you can create are endless.

True cross-platform development

With the click of a button, you can build applications for Macintosh (Classic and OS X), Windows, Linux, and Unix.

Increased productivity

The powerful interface builder and easy-to-use programming language speed up all the essentials of software development, leaving you with more time to focus on your project.

Databases, multimedia, Internet applications, external libraries and more

Revolution supports everything you need: SQL databases, streaming media, control of QuickTime, QTVR and graphics, CGI processing, Internet protocols, and more.

FREE Trial Version

www.runrev.com



MacUser UK - 5 mice

©2002 Runtime Revolution Limited. All rights reserved. Runtime Revolution, the Runtime Revolution logo and Revolution are trademarks of Runtime Revolution Limited, registered in the United Kingdom. All other trademarks are the property of their respective owners.



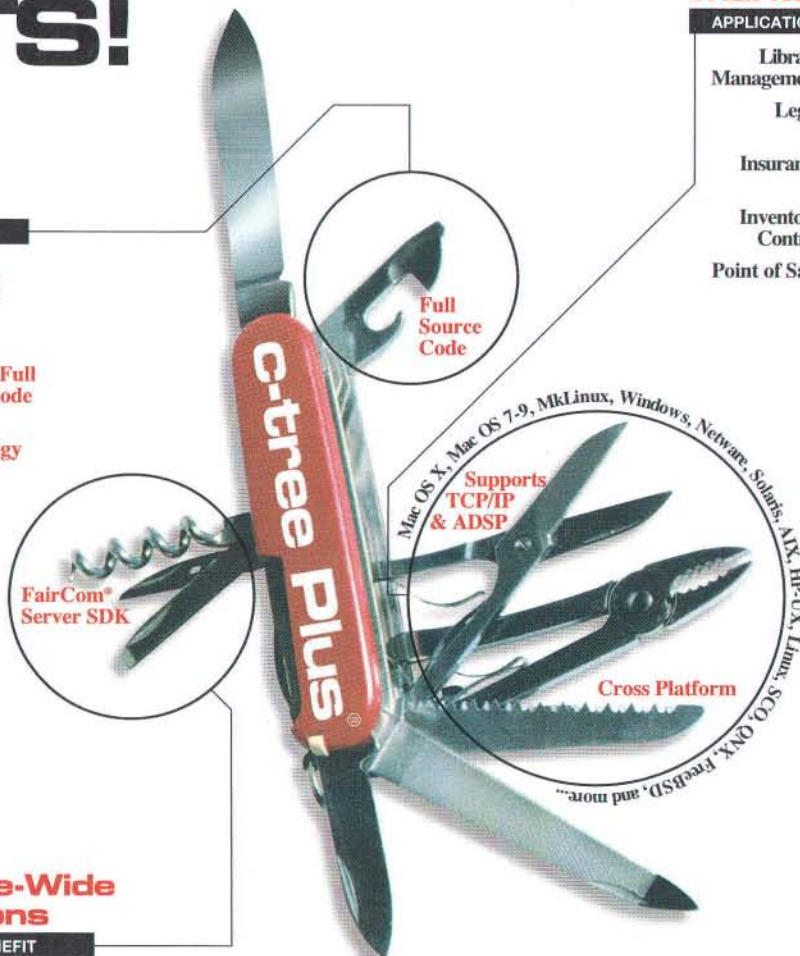
CUT YOUR DATABASE COSTS!

Embedded Hardware

APPLICATION	BENEFIT
Internet Appliances	Small footprint
Medical Devices	Stable
Factory Automation	Includes Full Source Code
Space Exploration	Proven Technology

Vertical Markets

APPLICATION	BENEFIT
Library Management	Easy Deployment
Legal	Flexible Licensing
Insurance	Huge File Support
Inventory Control	Scalable
Point of Sale	Cross Platform



Corporate-Wide Applications

APPLICATION	BENEFIT
B2B Server	Flexible Communication
Proprietary Applications	Fast
Dedicated Web Server	Robust Threading

C-TREE DELIVERS FAST, SOLID, AND FLEXIBLE DATABASE TECHNOLOGY WHILE CUTTING YOUR DEPLOYMENT COSTS...

For over twenty years, FairCom's proven database technology has been a favorite of commercial developers for Macintosh applications ranging from low level embedded appliances to vertical market applications to huge multi-platform, corporate applications. FairCom has established this reputation by delivering uncompromising Standalone and Client/Server technology in a flexible package that enables you to cut costs while utilizing cutting edge technology like our new HUGE file support and 64-bit support. FairCom offers this exceptional performance, unsurpassed data availability and rock solid reliability with a low total cost of ownership and flexible licensing options.

Get more for your development dollar! Visit www.faircom.com/ep/freecdooffer today for a free developer's CD.



FairCom
CORPORATION

FairCom

Offices

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

Mac Support Since 1985 • www.faircom.com • USA. 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

<http://www.mactech.com>

The MacTech Editorial Staff

Publisher • Neil Ticktin

Managing Editor • Jessica Stubblefield

Online Editor • Jeff Clites

Regular Columnists

Getting Started – Networking

by John C. Welch

Programmer's Challenge

by Bob Boonstra

MacTech Online

by Jeff Clites

Regular Contributors

Vicki Brown, Andrew Stone,
Tim Monroe, Erick Tejkowski,
Kas Thomas, Will Porter,
Paul E. Sevinç, Tom Djajadiningrat,
and Jordan Dea-Mattson

MacTech's Board of Advisors

Dave Mark, Jordan Dea-Mattson,
Jim Straus, Jon Wiederspan,
and Eric Gundrum

MacTech's Contributing Editors

- Michael Brian Bentley
- Derek J. Burks
- Tantek Çelik, Microsoft Corporation
- Marshall Clow
- John. C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Cobalt Networks
- Lorca Hanns, San Francisco State University
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Scott Knaster, Microsoft
- Peter N. Lewis, Stairways Software
- Rich Morin
- John O'Fallon, Maxum Development
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Andrew C. Stone, www.stone.com
- Michael Swan, Neon Software
- Chuck Von Rospach, Plaidworks
- Bill von Hagen
- Eric Zelenka

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • W2 Graphics

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Events Manager • Susan M. Worley

Network Administrator • Deryck Richards

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane

Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2002 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-1, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

June 2002 • Volume 18, Issue 06

SCRIPTING

- 6 **Shell Scripts, Perl and Python**
by Paul Ammann

COCOA DEVELOPMENT

- 22 **The Beauty of Categories**
Use This Objective-C Feature to Make Your Cocoa Code Cleaner
by Dan Wood, Alameda CA



Progress Bar at the bottom of the tilepage 66

SECTION 7

- 4 **Some Basic Commands**
by Rich Morin

PROGRAMMER'S CHALLENGE

- 27 **MATCHSTICKS**
by Bob Boonstra

COVER STORY

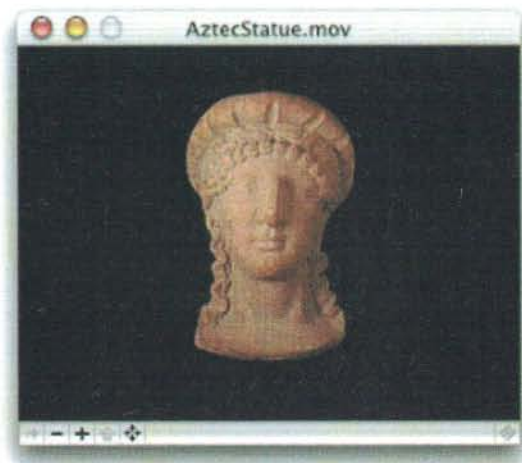
- 66 **Change your Image!**
Dock Tile Imaging
By Andrew Stone S. Downs

NETWORK MANAGEMENT

- 12 **Heterogeneous Networks as a defense mechanism**
Why a genetically diverse network has advantages that a network of clones can't match
by John C. Welch

QUICKTIME TOOLKIT

- 72 **Virtuosity**
Programming with QuickTime VR
by Tim Monroe



A QuickTime VR object moviepage 18

By Rich Morin

Some Basic Commands

Learn these first; the others can wait!

Before we look at any specific commands, we need to look at some fundamental differences between BSD terminal sessions and the ordinary Mac OS X (OSX) experience. Each interface has its advantages and disadvantages; by understanding these, you'll be able to use both interfaces more effectively!

- **Applications vs. Commands** OSX applications tend to be highly interactive, providing customized environments for accomplishing particular tasks. BSD commands, in contrast, are generally run in "batch mode". The user issues the command, giving all of the needed information, waits for the result, and goes on.
- **Documents vs. Files** OSX documents are generally encoded in an application-specific data format. Consequently, it is only by extra effort that other applications can make use of them. BSD files are expected to be useful to a wide range of commands. As a result, they are often formatted as ASCII text files, using white space (blanks, spaces, and newlines) to delineate internal data elements.
- **Subjects vs. Verbs** In OSX, the user double-clicks on the icon for a document (subject) and the appropriate application (verb) starts up. Alternatively, s/he drags the document icon (subject) over to the application icon (verb). In BSD, the user types in the name of a command (verb), followed by assorted file names (subjects) and flags (adverbs).
- **Mice vs. Keyboards** OSX employs a graphical user interface (GUI), using the mouse (trackball, ...) for

nearly every action. Keyboard "shortcuts" are provided in some cases, but they are never required. In BSD, the situation is exactly reversed. The keyboard is used for everything; mouse "shortcuts" are used only occasionally.

- **Windows vs. Sessions** OSX applications often put up multiple windows; actions taken in one window are expected to have ramifications in all of the other windows. Each BSD terminal session, in contrast, is expected to be independent of all of the other sessions.
- **Aqua vs. The Shell** Aqua (including the Dock and the Finder) allows the user to navigate through the file system, manipulate documents and folders, and start up commands. The BSD "shell" (assisted by some common commands) provides equivalent capabilities, plus others (e.g., scripting, session logging). Of course, the user interfaces vary substantially!

NAVIGATION COMMANDS

As noted above, BSD command lines start with a verb, followed by some number of subjects and adverbs. Last month, we used the "man" command to view manual pages; let's use it again, trying out some variations.

```
[localhost:~] rdm% man 2 sync
[localhost:~] rdm% man 8 sync
[localhost:~] rdm% man sync
```

The `sync(2)` manual page describes a system call; `sync(8)`, in contrast, is a system maintenance command. By specifying the section number, we can tell `man` which part of the manual to search. In most cases, there is no conflict, so the section number can be left off.

Rich Morin has been using computers since 1970, Unix since 1986, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.

None of **man**'s "arguments" are actually file names. Instead, they are hints that allow **man** to search assorted directories. The actual list of directories that **man** examines is specified by **MANPATH**, a BSD "environment variable". Environment variables and control files perform many of the functions of OSX preferences:

```
[localhost:~] rdm% echo $MANPATH
/Users/rdm/man:/usr/local/share/man:/usr/share/man
```

This tells us that **man** looks first under **/Users/rdm/man** (the user's personal man pages), **/usr/local/share/man** (this machine's "local" man pages), and **/usr/share/man** (OSX man pages). Let's wander over to the latter directory and take a look:

```
[localhost:~] rdm% cd /usr/share/man
[localhost:/usr/share/man] rdm% ls -F
man1/      man3/      man5/      man7/
whatis.db
man2/      man4/      man6/      man8/
```

The **cd(1)** command sets **/usr/share/man** as the "current directory" for this terminal session. This can save a *lot* of typing! For convenience, the shell puts the name of the current directory in the command line's "prompt string".

The **ls(1)** command provides a directory listing, appending slashes to directory names (as directed by the **-F** option). So far, the Finder seems a lot more convenient, but hang on a bit! Let's find out which directories have "sync" man pages:

```
[localhost:/usr/share/man] rdm% ls */sync*
man2/sync.2  man8/sync.8
```

The asterisk "wild cards" tell the shell to look through every subdirectory, looking for files whose names begin with "sync". Hmm; looks like the shell has some advantages when lots of items are involved. How many pages might that be, anyway?

```
[localhost:/usr/share/man] rdm% ls */* | wc -l
2853
```

This command "pipeline" ran two commands, directing the "standard output" of one into the "standard input" of the other. The first command listed every file in every subdirectory; the second counted the number of lines in the first command's output. The result (2853) was something which neither base command was "designed" to produce.

This pipeline is actually a minuscule instance of a "shell script". BSD users frequently write scripts to automate repetitive tasks. Next month, we'll look at some more commands and some fancier ways of writing shell scripts.



Fetch

Good doggie.

Fetchsoftworks.com

Version 4.0.2 now available.

By Paul Ammann

Shell Scripts, Perl and Python

In the early days computers were used to run programs, and nothing more. You punched your program on cards, delivered the pack to the computer department. The staff there loaded the program into the computer, executed it and retrieved the result on paper. This was in turn returned to you and you had to sit down and figure out why you got the result you got.

Modern computers are a little more complex than that. You have a complete environment where you can execute your programs and even have such astonishing things as *interactive* programs. It is no longer enough to be able to load your program and just print the result. You also need support to reformat the results, process them in other manners (maybe printing a nice diagram) and store them in a database. It would of course be possible to write specially designed programs that formatted the output of your programs according to your wishes, but the number of specialized programs would quickly increase, leaving your computer loaded with "might come in handy" programs.

A better approach would be to have a small set of processing programs with a program made to "glue the parts together." On a UNIX system, such a program is called the *shell*. The shell is used to issue commands, start processes, control jobs, redirect input and output, and other mundane things that you do on a modern computer. Not only that, the shell is a pretty complete programming language.

Now that Apple has switched to UNIX-based operating system, there are several new languages users can utilize: the Shell script, Perl, and Python. In this article we introduce each of languages, provides some insightful information about each one, and point out useful resources for readers.

THE BASIC SHELLS

The basic UNIX shells come in three main language forms. These are and in order of creation: Bourne Shell, C Shell, and the Korn Shell. Be aware that there are several dialects of these script languages, which tend to make them all slightly platform specific. The different dialects are due, in the main, to the different UNIX flavors in use on some platforms.

All script languages though have at their heart a common core, which if used correctly will guarantee portability.

Why use Shells? There are three reasons: (1) It is the simplest way to string a bunch of UNIX commands for execution at any time without the need for prior compilation; (2) It's generally fast to get a script going, not forgetting the ease with which other scripters can read the code and understand what is happening; and (3) they are generally completely portable across the whole UNIX world, as long as they have been written to a common standard.

The Bourne Shell

Historically the Bourne shell language, or **sh**, was the first to be created. It has a very compact syntax, which makes it obtuse for novice users but very efficient when used by experts. It also contains some power constructs built in. On UNIX systems, most of the scripts used to start and configure the operating system are written in the Bourne shell. It has been around for so long that it is virtually bug free.

The C Shell

Next up is the C Shell, or **csh**, so called because of the similar syntactical structures to the C language. The UNIX man pages contain almost twice as much information for the C Shell as the pages for the Bourne shell, leading most users to believe that it is twice as good. This is a shame because there are several compromises within the C shell, which makes using the language for *serious* work difficult (check the list of bugs at the end of the man pages!). True, there are so many functions available within the C Shell that if one should fail another could be found. The point is do you *really* want to spend your time finding all the alternative ways of doing the same thing just to keep yourself out of trouble. The real reason why the C Shell is so popular is that it is usually selected as the default login shell for most users. The features that guarantee its continued use in this area are aliases, and history lists.

Paul Ammann works as Network Security Engineer for a logistics company in Connecticut. He has been working with UNIX and Linux for the last 5 years. When not working, he is busy working on his Nokia Firewall Manager project, or plotting the next vacation adventure with his wife Eve. He can be reached at amani@users.sourceforge.net.

Application Builder Collection™

Reusable Cocoa frameworks for building solid Mac OS X applications

- ▶ Port to Cocoa faster
- ▶ Save thousands by reducing coding and testing time
- ▶ Spend your time developing the features that make your software stand out
- ▶ Quickly create professional looking applications
- ▶ Utilize object-oriented technology for easy customization and reuse
- ▶ No deployment licenses or royalties!

Collection Includes:

- ▶ Rotational Slider
- ▶ Formatters for Currency, Phone, Text, and Custom Fields
- ▶ Draggable Cocoa Cells for Tables
- ▶ Calendar View and Popup
- ▶ URL Clickable Field
- ▶ Twiddle View
- ▶ Archive Builder Application
- ▶ And More . . .

Learn more at www.jiiva.com



rotate images and more



control data input and presentation



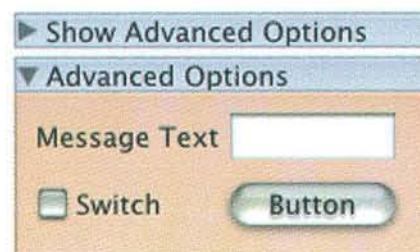
drag sliders and other widgets into tables



add calendar popups and views

Visit Jiiva's [web site](http://www.jiiva.com/web site)

turn text fields into clickable URLs



create complex interfaces with versatile twiddles



 **Jiiva**, Inc.
www.jiiva.com/abc

The Korne Shell

Lastly, we come to the Korne Shell, or **ksh**, made famous by IBM's AIX flavor of UNIX. The Korne shell can be thought of as a superset of the Bourne shell as it contains the whole of the Bourne shell world within its own syntax rules. The extensions over and above the Bourne shell exceed even the level of functionality available within the C shell (but without any of the compromises!), making it the obvious language of choice for real scripters.

What is a Shell Script Anyway?

A shell script, in its most basic form, is simply a collection of operating system commands put into a text file in the order they are needed for execution. Using any of the shell mentioned so far, a text file containing the commands listed in Example 1 would work every time the script was executed.

Code 1. Basic shell script

```
#!/bin/sh
rm -f /tmp/listing.tmp > /dev/null 2>&1
touch /tmp/listing.tmp
ls -l [a-z]*.doc | sort > /tmp/listing.tmp
lpr -Ppostscript_l /tmp/listing.tmp
rm -f /tmp/listing.tmp
```

Of course not all scripts are this simple but it shows that ordinary UNIX commands can be used without any extra, fancy scripting constructs. If this script was executed any number of times the result would be the same, a long listing of all the files starting with lower case letters and ending with a **doc** extension from the current directory printed on your local PostScript printer.

If Shell scripting isn't for you, there are two other languages to consider: Perl and Python. Both languages are similar in functionality. However, in terms of syntax, they are as different as night and day. In the next section, we'll take a close look at both of these languages, and develop a UNIX daemon that will demonstrate the same functionality and the contrast of syntax.

PERL VS. PYTHON

Perl is a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical (i.e., easy to use, efficient, complete) rather than beautiful.

Perl combines some of the best features of C, **sed**, **awk**, and **sh**, so people familiar with those languages should have little difficulty with it. Expression syntax corresponds closely to C expression syntax. Unlike most UNIX utilities, Perl does not arbitrarily limit the size of your data—if you have the memory, Perl can slurp in your whole file as a single string. Recursion is of unlimited depth. In addition, the tables used by hashes grow as necessary to prevent degraded performance. Perl can use sophisticated pattern matching techniques to scan large amounts of data quickly.

On the other hand, Python is simple to use, and it's more of a real programming language, offering much more structure and support for large programs. It also offers much more error checking than C, and, being a *very high-level language*, it has high-level data types built in, such as flexible arrays and dictionaries that would cost you days to implement efficiently in C. Because of its more general data types, Python is applicable to much larger problem domain than **awk** or even Perl.

Python allows writing very compact readable programs. This is due to the fact that statement grouping is done by indentation instead of begin/end brackets. Like Perl, Python is an interpreted language; however, Python is an object-oriented language like C++ but without the headaches of constructors or destructors.

I could go on and on about Perl and Python. However, I'll let the code speak for itself.

Writing a UNIX Daemon in Perl and Python

The word *daemon* is derived from the Greek word *daimon*, meaning “a supernatural being” or “spirit”, rather than demon, referring to the fallen angels or followers of Satan. Some would insist that UNIX is infested with both daemons and demons. In UNIX, daemons are typically started by the root process when the operating system is initialized, and run in the background indefinitely. Daemons typically spend most of their time waiting for an event or period when they will perform some task.

UNIX Processes

Before we explore the details of what a daemon is, let's review the characteristics of a UNIX process. First, let's take a look at a partial process list:

```
> ps aux
PPID  PID  PGID  SID  TTY      TPGID  STAT  UID   TIME
COMMAND
0      1      0      0  ?        -1  S      0     0:03
init
1      2      1      1  ?        -1  SW      0     0:00
[kflushd]
1      3      1      1  ?        -1  SW      0     0:00
[kpiod]
1      4      1      1  ?        -1  SW      0     0:00
[kswapd]
1     548    548    548  ?        -1  S      0     0:00
httpd
548    557    548    548  ?        -1  S     99     0:00
httpd
548    558    548    548  ?        -1  S     99     0:00
httpd
548    559    548    548  ?        -1  S     99     0:00
httpd
548    560    548    548  ?        -1  S     99     0:00
httpd
548    561    548    548  ?        -1  S     99     0:00
httpd
548    562    548    548  ?        -1  S     99     0:00
httpd
548    563    548    548  ?        -1  S     99     0:00
httpd
2450   2452   2452   2452 pts/5    2576  S     501    0:00
zsh
```

The first column is the parent process id, the process id of the process that created it. The second column contains the process id, which is assigned by the kernel. The next column is the process group id. The fourth column is the session id. A session is a collection of session groups. The next column contains the tty, or controlling terminal, that's related to the process. This is typically a terminal or remote login shell.

Daemon Rules

There are several rules or characteristics that most daemons possess. If a daemon does not follow these basic rules, it will most likely become a demon and wreak havoc on your system.

Fork and Exit

The first thing a daemon should do is **fork()** a child process and have the parent process **exit()**. This is necessary for several reasons. First, it disassociates the process from the controlling terminal, or the login shell. Second, it removes the process from the process group that initiated the program. This ensures that the process is not a process group leader, which is required for **setsid()** to run successfully.

Call setid()

setsid() is a POSIX function that turns the process into a session leader, group leader, and ensures that it doesn't have a controlling terminal.

Change Working Directory

The current working directory should be changed with **chdir** to the root directory (/). This is necessary to avoid using a working directory that resides on a mounted partition. If the process is started on a mounted partition, the system administrator wouldn't be able to unmount the partition until the process was halted. You could specify a different working directory as long as it doesn't reside on a remotely mounted partition.

File Creation Mode

The **umask** determines the default permissions new files are assigned. Setting **umask** to 0 removes the defaults that might otherwise disable permissions the daemon intended to enable when creating files.

Close Unneeded File Handles

Besides closing any filehandles that might have been opened, daemons often redirect STDIN to read from, and STDOUT and STDERR to write to /dev/null.

Code Example 2. Redirecting STDIN, STDOUT and STDERR in Perl

```
open STDIN, '/dev/null';
open STDOUT, '>/dev/null';
open STDERR, '>/dev/null';
```

Code Example 3. Redirecting STDIN, STDOUT and STDERR in Python

```
sys.stdout = open("/dev/null", 'w')
sys.stderr = open("/dev/null", 'w')
sys.stdin = open("/dev/null", 'r')
```

5 times faster

NEW: PRIMEBASE 4.0



Using PrimeBase, you are able to produce Internet and Intranet Applications faster and with the best performance ever.

The development package includes the

- PrimeBase Application Server
- PrimeBase Development Framework
- PrimeBase SQL Database Server

[get developer key
free of charge]

download now!
www.primebase.com

Develop on a Mac and deploy without any additional effort on any platform you want.

- Completely cross-platform
- Full-text searching and indexing
- Object-oriented
- Separation of code and layout
- Native connectivity to any database



PRIMEBASE

SNAP Innovation GmbH

Altonaer Poststraße 9a

D-22767 Hamburg / Germany

www.primebase.com

e-mail: info@primebase.com

Fon: ++49 (40) 389 044-0

Fax: ++49 (40) 389 044-44

Logging Messages

In some cases, you may want to record an error or message to a log file on the system. This can be accomplished by redirecting STDOUT and STDERR to one or more log files.

Code Example 4. Redirecting STDOUT and STDERR to one or more log files in Perl

```
open STDOUT, ">>$access_log" or die "Can't write to
$access_log: $!";
open STDERR, ">>$error_log" or die "Can't write to
$error_log: $!";
```

Code Example 5. Redirecting STDOUT and STDERR to one or more log files in Python

```
try:
    sys.stdout = open(">>access_log", 'w')
except:
    print "Can't write to access_log"
try:
    sys.stderr = open(">>error_log", 'w')
except:
    print "Can't write to error_log"
```

Writing the Daemon

Now that we understand the basic attributes of a daemon, let's put the pieces together into a simple Perl/Python program.

Code Example 6. Simple Daemon in Perl

```
#!/usr/bin/env perl

# load required modules
use strict;
use POSIX qw(setsid);

chdir '/' or die "Can't chdir to /: $!";
umask 0;
open STDIN, '/dev/null' or die "Can't read /dev/null:
$!";
# open STDOUT, '/dev/null' or die "Can't write to
/dev/null: $!";
open STDERR, '/dev/null' or die "Can't write to
/dev/null: $!";
defined(my $pid = fork) or die "Can't fork: $!";
exit if $pid;
setsid or die "Can't start a new
session: $!";

while(1) {
    sleep(5);
    print "Hello...\n";
}
```

Code Example 7. Simple Daemon in Python

```
#!/usr/bin/env python

import os, sys, time

os.chdir("/")
os.umask(0)

try:
    sys.stdout = open("/dev/null", 'w')
except:
    print "Can't read /dev/null"
try:
    # sys.stderr = open("/dev/null", 'w')
    pass
except:
    print "Can't write to /dev/null"
try:
    sys.stdin = open("/dev/null", 'r')
except:
    print "Can't write to /dev/null"
```

```
try:
    pid = os.fork()
except:
    print "Can't fork"
if pid:
    sys.exit(1)

try:
    os.setsid()
except:
    print "Can't start a new session"

while (1):
    time.sleep(5)
    print "Hello..."
```

Code Examples 6 and 7 are a simple daemon that will print **Hello...** to the console every five seconds. Also, looking at both examples, notice that the line of the program has been commented out. Removing the comment will suppress the hello message by sending all standard output to `/dev/null`. Also, notice that we included the POSIX library and explicitly imported the **setsid** function since this function is part of the POSIX library, not a built-in Perl function. One other little critical piece is the **while(1) { }** loop with **sleep(5)** inside. The loop ensures that the script will run indefinitely. The sleep function sets the number of seconds between each iteration in the loop. The main body of your code will sit inside this while loop.

In this brief tutorial, we learned how to create a robust daemon in both Perl and Python for most any UNIX system. Which language is better? I don't know. Perl has been around since 1993 and there is a treasure chest of source code available for it. However, there is nothing to stop you from porting Perl code over to Python, just a few minor changes.

RESOURCES

For readers who are interested in learning more about Shell programming, I can highly recommend *UNIX Shell Programming* by Stephen G. Kochan and Patrick H Wood. I keep one copy at home, and one at work. In addition, check out Heiner Steven's web site SHELLdorado (<http://www.shelldorado.com/>) and Sun Microsystems has a very resourceful web site (<http://www.sun.com/bigadmin/scripts/index.html>).

If you are interested in getting your feet with Perl, *Learning Perl* by Randal L. Schwartz and Tom Phoenix is my pick. By the time you've finished this book, you will know if Perl is the right language for you without making a huge investment. For me, *Core Python Programming* by Wesley J. Chun was the book that converted me from Perl. Enough said.

Regardless if you choose Perl or Python, check out Active State, which make IDE programs for both languages and has a large library of code for beginners (<http://aspn.activestate.com/ASPN/Cookbook/>). If you decide to write a major application with either, I would recommend wxPerl (<http://wxperl.sourceforge.net/>) and wxPython (<http://wxpython.org/>). Both software libraries are derivatives of wxWindows (<http://www.wxwindows.org/>), which is a free C++ framework to make cross-platform programming child's play. Well, almost. Enjoy!

Get your
**software sales
ripping!**

NEW VERSION!
Optimized for MAC OS X



Privilege®

Tear up that road to market! Allow customers to download your software via the Internet — faster and safer — with the Privilege Software Commerce Solution. Secure trial versions and flexible licensing models that increase your revenue while you simultaneously decrease your operational costs.

To learn why five of the six leading publishers have chosen Aladdin's Privilege to protect, distribute and sell their software over the Internet — zip over to **eAladdin.com/Privilege**. For the full blazin' story, call **1-800-562-2543** and start ripping!

Aladdin®
SECURING THE GLOBAL VILLAGE
eAladdin.com

By John C. Welch

Heterogeneous Networks as a Defense Mechanism

Why a genetically diverse network has advantages that a network of clones can't match

WELCOME

Crackers. Viral attacks. Inside attacks. The computing world is fraught with peril these days. Yet as much as we would like to hide away from these things or ignore them, we have to deal with computers, computer networks, and the security and privacy issues they bring. But the question is always, how do we counter all the bad things in the computing world so that we can get to work? There are quite a few methods, some more common than others, none of which are a panacea. The most used methods are prophylactic, or external defenses/protections, and while they have achieved a certain amount of success, such measures, in the end, guarantee a certain amount of failure as well.

EXTERNAL DEFENSES AND ASSOCIATED ISSUES

The way most people defend their networks and computers are by creating barriers to external attacks. For example, to defend against virii, we buy antivirus applications. We use firewalls to prevent crackers from gaining access. We use arcane password requirements and policies, and stringent policing of internal network usage to keep employees and other authorized users from causing damage. More concisely, we use prophylactic hardware, software, etc. Now, I'm not going to say these are bad, or unnecessary actions. In fact, they are quite intelligent actions. But, as we will see, on their own, they are not ever going to be enough.

PROBLEMS

Virii

Virus defense is a particularly thorny issue. In the last few years, you've gone from only having to update viruses monthly, to daily updates. You have to scan almost every file that you download or use, and you have to repeatedly scan the same files,

because virus writers get smarter all the time. Viruses are also becoming smarter, or at least more flexible in design and execution. They change their code each time they run, they dance from sector to sector of the drive, some will even dodge virus scans.

It's not even an operating system issue any more. Application - based virii have surpassed operating system virii in number and infection rates. It's also not just a Microsoft application issue. QuickTime, Adobe Acrobat, and other applications from different vendors are being infected. Services are getting slammed the same way. IIS, and other Web/Internet services are being used as viral delivery mechanisms. While things like IIS attacks are limited to a single operating system, the way services like IIS work means that the virii that target them can also hurt non-Windows services by creating a denial of service, (DOS) attack situation as they probe Apache, iPlanet, WebSTAR, and other Web Service Servers looking for a valid entry point.

Unfortunately, the image of a virus creator as some demented savant with no social skills sitting alone in the dark wreaking havoc on a cruel world is absolutely incorrect. There are now virus 'kits' on the Internet that allow anyone who can work a GUI to create virii that are quite advanced in nature while possessing only minimal technical abilities. The 'Kournikova' virus is an example. The creator of this Microsoft Outlook virus was not a cracker, or any other kind of programmer. He wasn't a computer expert of any kind. He was an Anna Kournikova fan who wanted to spread her fame. The method he chose, while damaging and annoying, was the modern version of Led Zeppelin fans in the 1970s spray - painting 'Zof'o' on bridges and highway overpasses. So almost anyone with Internet access can create viruses.

While antivirus vendors have done an admirable job, with on the fly scanners, email attachment scanning, server scanning, heuristic analysis, (which looks for virus - like behavior as opposed to static 'signatures'), there is still a problem with relying on antivirus utilities as the sole form of protection. Delay. There is always a delay between the discovery of the virus and the release of the inoculation definitions. As well, there is a delay between the announcement of the release and the downloading and vaccination of infected systems. There is also the delays in

John Welch <jwelch@mit.edu> is the IT manager for the MIT Police department, and the Chief Know-It-All for TackyShirt. He has over fifteen years of experience at making computers work. John specializes in figuring out ways in which to make the Mac do what nobody thinks it can, showing that the Mac is the superior administrative platform, and teaching others how to use it in interesting, if sometimes frightening ways. He also does things that don't involve computertry on occasion, or at least that's the rumor.

○ The World Is Flat.
○ Man Will Never Fly.
○ Database Independence Implies Poor Performance.

○ Another Myth Laid To Rest.

It has been a long held theory that database independent data access (ODBC, JDBC, OLE DB) provided inferior performance to database specific middleware (Native Drivers), thereby providing justification for the acquisition of costly and inflexible database specific solutions to serve your internal IS and external solutions.

As the leading technology innovator and provider of Database Independent Data Access Middleware since 1992, OpenLink Software has consistently developed, deployed, and supported **High-Performance ODBC, JDBC, and OLE DB Data providers** that alleviate the pains associated with Database Specific Middleware and refute the claims associated with PERFORMANCE. **Empower your enterprise today by downloading a copy of the OpenLink Universal Data Access Drivers Suite from our Web site or call us at 781 273 0900 for more information.**

www.openlinksw.com

➔ **PLATFORM SUPPORT**

MacOS & MacOS X, Windows 95/98/NT/2000/XP, Linux, Unix (all flavors)

➔ **DATABASE ENGINE SUPPORT FOR:**

Progress, Oracle, Microsoft SQL Server, DB/2, Informix, Sybase, CA-Ingres and other ODBC compliant database engines



CONNECT TO REMOTE DATABASE ENGINES WITH OR WITHOUT ADDITIONAL DATABASE VENDOR PROVIDED NETWORKING MIDDLEWARE

 **OPENLINK**
SOFTWARE
Universal Data Access Without Boundaries.

release times between different antivirus vendors of new definitions. In the case of a virus like Code Red, which spread at amazingly high speeds, this means that network administrators have to shut down vulnerable systems until the systems can be cleaned. With Code Red, this meant shutting down entire networks. Clearly, the measure - countermeasure - counter-countermeasure dance of virus and antivirus is not a winning strategy. (This should in *no* way be interpreted as saying that antiviral utilities are not worth the effort. Any security and protection plan should be layered, and antiviral utilities are a critical part of at least one of those layers. Even on platforms with a normally low rate of infection, if there is an antiviral utility available, and you aren't using it because 'you never see <my platform> being infected, that just happens to <some other platform>', then you are, to put it nicely, deluding yourself.)

Security and crackers

Non-viral attack points are everywhere as well. The recent announcements of vulnerabilities in SNMP and PHP affect any operating system that can run these services, (which is almost all of them, and in the case of SNMP, includes non - Mac OS X Macs.) Security holes are endemic on all platforms, even if they don't get the same level of publicity as the Microsoft security holes.

While all of these holes can be patched, there is the same delay as with virii. The vendor has to be notified of vulnerabilities, the vulnerability has to be analyzed, fixed, the fix tested, and distributed. In some cases, the patch creates other problems which have to go through the notification-analysis-fix-fix test-fix distribution cycle. Meanwhile, crackers are able to use these holes to break into systems and at very least, increase an already overburdened IT department's workload, which almost ensures that mistakes will be made in patch implementation, creating situations where systems that aren't patched are thought to be patched, and are now 'stealth holes' into a network.

A major problem with defending against crackers in this manner is one that you can't patch or secure against, and that is the attitude that the non-IT world takes towards them. For the most part, they are either viewed as misguided, or crusaders helping overworked, or (more commonly) inept IT people secure their networks. Because people don't think of computer data as 'real' and time has never been seen to have an inherent cost in IT, since we are all paid on salary, the time that these criminals take from IT people is seen as the IT person's just reward for not doing their jobs correctly. One cracker, Adrian Lamo, has gained a measure of fame as a 'security crusader' by breaking into systems, changing data, then informing the owners of the system of his attack, then offering his services to fix the holes.

Now, I think that any intelligent network administrator will, at some point, hire someone to try and penetrate their security. That is the only way to ensure that you have done things correctly. But the important thing here is that the people running the attacks have been *hired* for that purpose. If someone cracks your network, without being authorized to do so, then there is a more precise term for their actions: Breaking and Entering.

I am completely serious about this. If you came home, and found someone sitting in your Barcalounger watching your TV, drinking your beer, and they told you that they had broken into your home to show you how bad your security is, and they'll happily fix that for you if you hire them as a security consultant, you would first laugh at them, then call the cops. There is no difference save the physicality of the house break in. But because computers aren't 'real' then breaking into one isn't a 'crime' in most people's eyes. That attitude needs to be changed. Professionals don't break into my systems if they want me to hire them. They come to me and offer their services. If I accept, then they try to find my security holes, and help me close them. If I don't accept, then they go away except for annoying sales calls. If I get a visitor telling me how they broke into my network, and offering their services so it can't happen again, the first thing I'm doing is calling the cops. The second thing I'm doing is hiring a professional security consultant to work with me to patch these holes. If nothing else, am I seriously supposed to professionally trust someone who uses a lack of ethics as a sales tool?

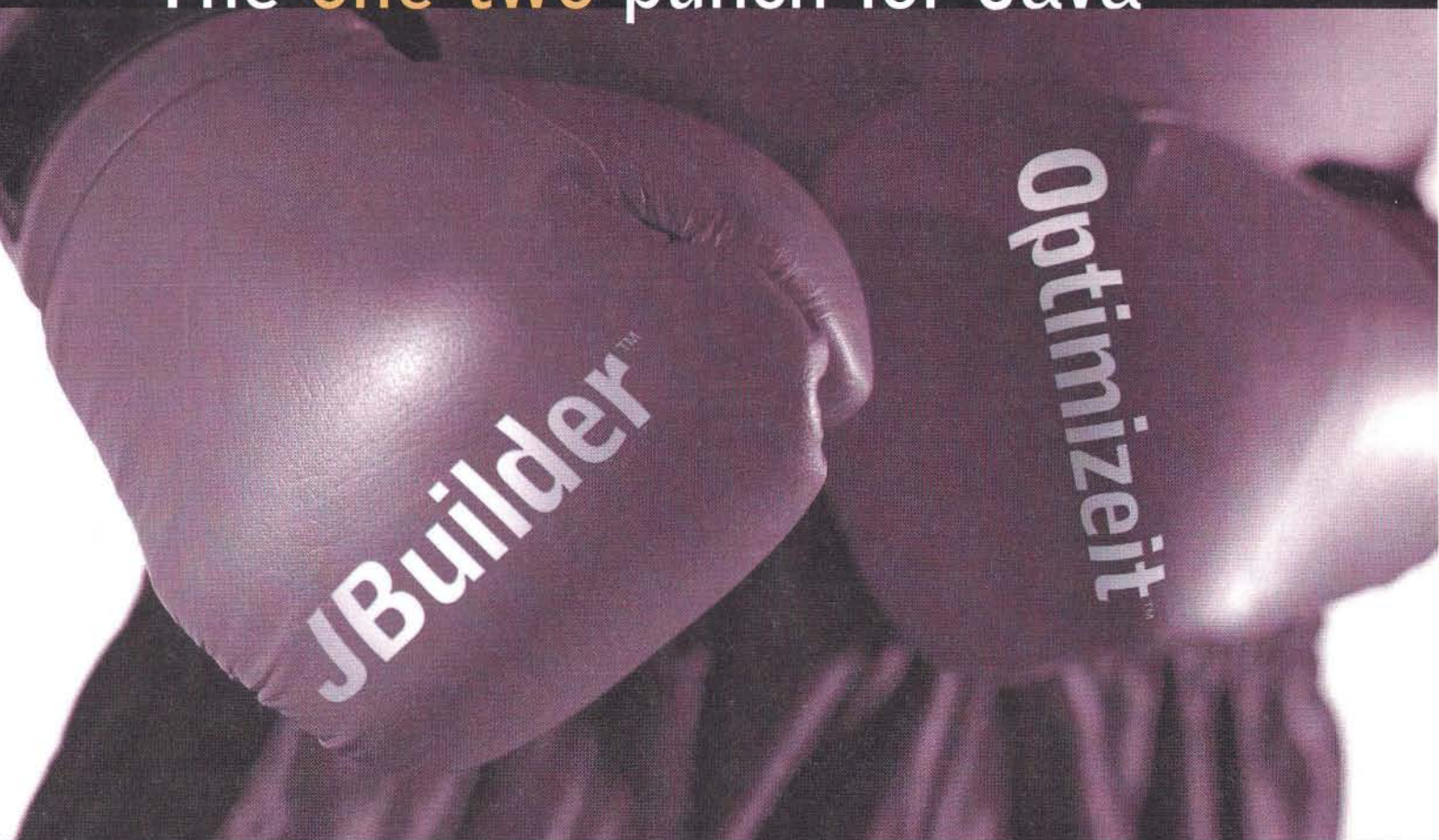
But increased security isn't the 'uber-fix' that people think it is either. In some cases, security can create it's own problems if applied incorrectly. Overzealous security policies applied without thought to their effects on the people that have to live with them can make daily use of computer resources so difficult that employees feel they have to find ways around them just to get work done. Email scanning for dangerous payloads is often used as an excuse to scan email content for 'inappropriate content', which is defined so nebulously that employees feel the need to use non-company email accounts while at work, which creates more security holes that have to be patched, and still more onerous security policies.

One of the sillier side effects of the viral problems that gets passed off as security is the severe restriction or even banning of email attachments. This is not just an IS policy issue. One of Microsoft's first anti-viral Outlook updates essentially made it impossible to use attachments with Outlook. Obviously this is not a sane answer. Attachments make email far more useful than it could otherwise be. Yes, they can be misused, but so can HTML email, and no one seems to be starting a trend to ban that. If you restrict email to the point where no one wants to use it, then you are just killing a critical tool to avoid a problem that you will end up with anyway.

Remember that in general, the more secure a system is, the harder it is to use. This is particularly evident when you see companies implementing physical security policies like removing floppy / CD drives, or placing locks on them so that you can't casually use them. While a company is perfectly in the legal right to do this, such a serious indication that the employees can't be trusted is never a good policy unless you have other security requirements that *require* you to do this.

So security is even more of a balancing act than antivirus defenses are. If you go too far, the system becomes unusable. If you don't go far enough, then you are under attack constantly, by your own machines in many cases.

The one-two punch for Java™



Powering Java development

A winning combination

Successful developers use Borland® JBuilder™ 7 and Borland Optimizeit™ Suite 4.2 together.

Build heavy-hitting Java applications

JBuilder is the leading environment for building Java applications. Rapidly develop Web Services and database applications. The highly productive JBuilder environment includes refactoring, unit testing, team development, and more.

Focus on lightning-fast, reliable code

The Optimizeit Suite is an essential for managing performance. Efficiently address speed, memory, and thread issues, so the applications you deploy will scale in production.

Borland®

Special price for the pair

For a limited time, get both JBuilder 7 and Optimizeit Suite 4.2 for a single, low price. For details, visit www.borland.com/new/jb7/5070.html or call 1-800-252-5547 (offer code: 5070)

Human issues

These are not only the most complex, but they will also cause you the most problems if not dealt with correctly. First, you have user training. If you do not train the people on the network correctly and hold them accountable to that training, then you don't have a prayer of any other external protections working. But there are inherent problems that crop up constantly with the user training solutions.

Training isn't cheap. External training can cost from a couple of hundred to a couple of thousand dollars per person per course. This can get prohibitively expensive for larger corporations. In house trainers may be cheaper, but what department wants to have to deal with that headache and expense. You still need facilities, equipment, lesson plans, courseware, study materials, etc. As well, the first budget to be cut, and the last to be restored is always the training budget. So what ends up happening is a new employee gets a list of 'Don'ts' that they read just enough to find the part where they are supposed to sign, acknowledging that they have indeed absorbed these important items into the very fiber of their being, hand it back to their boss, or HR, and then forget it ever existed. Training could be one of the best defenses against network break-ins and viral attacks, but not until it is seen as being as critical as power, water, and the CEO's bonus.

While eliminating training of the general user population seems as ignorant and short-sighted as it is, it pales to the way that most corporations treat the people tasked with keeping the network safe and running efficiently. IT departments will be told that they are critical to a company's operations, then get their budgets slashed due to reasons from the economy to the CEO hearing from a golf partner that outsourcing will save him more money than the company makes in a year. The IT staff has to deal with every facet of a network and all attached devices, yet they get no more of a training budget than anyone else, namely none. In addition, since they are usually looked at as a drain on the company's bottom line, their requests for additional funding get analyzed more than almost any other department.

IT departments are perennially short-staffed, even as their workload increases yearly, monthly, sometimes daily. Companies tell you up front that you get paid for 40 hours, but the minimum work week is 60 hours, with mandatory overtime. If you try to do your job in eight hours and go home, you are seen as 'not a team player', and let go at the first opportunity. The problem this creates is high turnover rates, with some companies replacing entire departments every couple of years when the numbers are averaged out. As a result, there is almost no 'institutional' memory in corporate IT departments, because that requires that your senior IT staffer not be the only one to have been there for over a year.

Ideally, the IT people will document what they do, so that when they are gone, there is a history of what has been done to and with the network. The reality is that when you are as overworked as most IT people are, documentation never even enters the picture, much less actually getting done. So every time an IT person leaves, that knowledge is gone, and has to be

relearned by the next person, who then leaves, and the cycle continues. Some companies are trying to do something about this, but they are too few, and still too far ahead of the curve for it to become standard practice.

What this means is that you cannot always rely on having an IT staff that is intimately familiar with your network, because chances are they either haven't been employed long enough to have achieved that level of knowledge, or are on the way out, and no longer care.

END RESULTS

The end results of these factors is that prophylactic protection, because of inherent implementation issues, useless training initiatives, and IT staff turnover, simply cannot work on their own. But there is another cause that accelerates these results, and it is genetic in nature.

NETWORK HOMOGENEITY IS A ROOT ENABLER FOR NETWORK VULNERABILITY

Almost any article on increasing your IT efficiency, improving your ROI, decreasing IT expenditures, making your network easier to manage and protect will eventually recommend that you take, among any other human or computer actions, the step of standardizing your computing platforms on a single client and server platform. I propose that regardless of what platform you settle on, that by making your network completely, or almost completely homogenous, that you are creating vulnerabilities that no amount of protection can fix.

The best examples of this are in the non-computing world. Human and animal diseases spread fastest when there is a degree of genetic uniformity among the species being attacked. It is well known among animal and plant breeders that any group that is too inbred is vulnerable to diseases and conditions than a group with a more diverse genetic background. The histories of plant infestations demonstrate how a single species can be nearly destroyed within a relatively short time by a single disease. The Irish famine of earlier centuries is a rather extreme example of this. The potato crops in Ireland were nearly destroyed by a blight, which, due to the almost total dependence on that plant by the Irish people, caused a massive famine.

Other examples of the ways that excessive genetic similarity creates problems are the conditions and diseases that only affect certain groups of people, or affect only certain groups in any kind of number, such as sickle-cell anemia. This example can easily be applied to network design and implementation.

If you have a network that is all, or mostly homogenous, then a new virus has a guaranteed population to infect and spread itself from. A homogenous network with high-speed connections, and heavy traffic spread virii at astounding rates before the infection is discovered and dealt with. Melissa, Code Red, Kournikova, and the rest are perfect examples of this. The world-wide-rate-of-Code Red infection should have been a wake up call, and it was, but only lately is it becoming the *right* kind of wake up call.

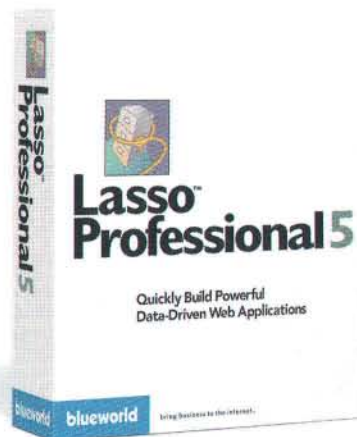
“Awesome!”

“The power, quality and feature set of Lasso Professional 5 is very impressive. The new documentation is definitely among the best in the business.”

Johan Sölve; Halmstad, Sweden

“Blue World has managed to add an immense amount of functionality and scalability without sacrificing any of the ease-of-use of previous versions. New features like LassoApps, custom tags and LassoScript create a development environment that seems almost limitless.”

Tom Wiebe; Vancouver, Canada



“With Lasso, I have been able to single-handedly develop sophisticated solutions for internal and external use in less time than I see teams of people take with other middleware languages.”

Greg Willits; Santa Ana, California

“This new release, I have got to admit, is truly amazing. The rich array of new features, the brand new documentation and the amazing new administration interface make Lasso Professional 5 definitely worthy of a purchase.”

Brian Olsen; Brooklyn, New York

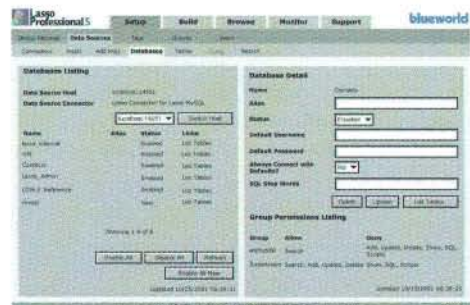
Upgrade today to the most powerful Web application server for Macintosh and beyond and you'll discover why so many Web developers claim Lasso Professional 5 is the must-have tool for quickly building and serving powerful data-driven Web sites.

Lasso Professional 5 introduces a next-generation, object-oriented Web programming language, advanced Web application server administration, an embedded Lasso MySQL™ high-performance database server, a new distributed architecture, new platform and data source support, unprecedented extensibility and customization, 1800 pages of rewritten documentation and over 200 new features and enhancements. Lasso Professional 5 provides vast new power and features while maintaining the legendary ease-of-use, performance, and reliability that make Lasso the preferred tool for tens of thousands of Web developers, ISPs, and IT/IS professionals.

If you're serious about building and serving data-driven Web sites, but don't want to spend a serious amount of time getting it all to work, there's no better choice than Lasso Professional 5.

Visit the Blue World Web site today and see how Lasso products can help you quickly build and serve powerful data-driven Web sites.

Lasso – The Leading Web Tools for Macintosh and Beyond



Lasso Administration controls your entire setup via an attractive and intuitive Web browser interface.



Lasso Database Browser provides instant access to all your databases, without writing a line of code.

blueworld
www.blueworld.com

No matter how good your defenses, if your network is nothing but a series of clones, then they all have exactly the same weaknesses to virii, or crackers. This has nothing to do with platform. An all - Sun, or all - Mac OS X network is just as vulnerable to a Sun or Mac OS X - specific attack as an all Windows network is to something like Code Red, or Melissa. Because the virii or attacker only has to deal with a single set of possible entry points, the job of the cracker or virus creator is greatly simplified. All they have to do is construct the virus or attack around a given platform's weaknesses, and they are assured of at least early success. If the people breaking into your network, or writing virii are actually talented or skilled, they can use that target specificity to make avoiding detection even easier. If all you have to deal with is Windows, or Solaris, or HP-UX, then you have a much better chance of avoiding detection simply because the conditions you have to deal with are drastically reduced. Popularity of platform, not quality of platform is why most virii center on Windows.

We don't, (for the most part), approve of cloning for humans or domesticated animals, so why do we not only approve of it for computing, but champion it as the answer to all our problems? If genetic homogeneity is a threat to well - being everywhere outside of computers, how can any intelligent person think that those problems magically disappear just because it's a computer? It's not just computers where this falls down. If the US Air Force consisted of nothing but F-16s and B-1Bs, then defeating the USAF in battle goes from an extremely difficult goal to one that is relatively simple. The level of homogeneity that exists in some computer networks would be thought of as either illegal or the height of stupidity in any other area, so why continue to use a method that is so obviously flawed?

Money spells it out quite well. By having single sources for hardware, software, and service, your up front costs for a computing environment are greatly reduced. Regardless of platform, if everything comes from one place, you save money, initially. Up front costs are also the most obvious. How do you show that you saved money because something didn't happen? You almost can't, short of letting the attack or infection happen, tallying the costs, and using that as justification for implementing diversity. While this would clearly show the hidden costs of over - homogenizing your network, the lack of ethics inherent in such an action would, and should get the people involved in such an action fired, if not arrested and sued. To put it bluntly, you cannot really show the cost of something that didn't happen. The best you can do is use the misfortune of others as your justification.

GENETIC DIVERSITY IN NETWORKS AS A STRENGTH AGAINST ATTACK

So, how do you go about implementing genetic diversity on your network? You have to correctly analyze your needs. Too many IT divisions get suckered into using the preferred platform as the basis for determining network implementation. If you look at everything from the standpoint of "How do we get <Windows/Solaris/Mac OS X/AIX> to fix this problem, you're

already doomed. The platform has to be determined by the solution, not vice - versa. (This is not to say that you will never have a need for limited amounts of homogeneity. If you have a group of people that are writing Windows drivers, then obviously they need to have Windows computers. Anything else would be inane.)

DEFINE THE PROBLEM CORRECTLY

Rather than think about it from the platform, think about the problem on its own. What is the problem? We need faster file servers? We need better network management? Define the problem on its own. The need for faster file servers has nothing to do with Windows or Linux, unless the current servers are running those operating systems. Even then, that should only be an historical note. At this stage, no platform should be excluded from consideration.

Advantages

The advantages of correct problem definition are numerous. First, you can avoid going for the quick solution that may only mask the problem. I have seen problems with speed that are really caused by an overburdened infrastructure get patched by adding servers so that each server has less work to do. The infrastructure is still overburdened, but the problem is hidden because the individual servers are more lightly loaded.

Another advantage is that you often find the problem is not nearly as complex as it initially seemed. For example, there was a company with a normally reliable 16Mbps Token Ring network that one day started to go down almost constantly with no apparent reason. One of the first proposals was to yank out all the Token Ring infrastructure and replace it with Ethernet. Luckily the expense that this solution entailed kept it from being immediately implemented. What the problem turned out to be were unlabeled jacks. It seems the company had recently upgraded the number of network drops, and each faceplate had two network drops, along with a phone jack, but the labeling of the jacks had been put off for last, so the jacks could be installed marginally faster. Both the phone system and the Token Ring connectors were RJ - type connectors, the phones using RJ-11 connectors, and the Token Ring cables using RJ-45 connectors. So, a single user, not realizing the difference, plugged the phone into the Token Ring port. There was just enough contact between the connectors in the plug and the cable so that the net result was the network was going down, seemingly without cause. The correct solution ended up being essentially free, and far less traumatic than a complete infrastructure replacement would have been.

So defining the problem correctly, without preconceived solutions is critical to correctly implementing a genetically diverse network.

ANALYZE EVERY POSSIBLE SOLUTION

So you have defined the problem. Next, see what the possible solutions are. Platform should still not be a factor here.

You need to look at all possible solutions for appropriateness that is not specific to any platform. In our file server speed solution, a possible solution may be to give all users 100GB UltraSCSI RAID stacks on their desks. However, this is impractical for many reasons, none of which have to do with the platform on the user's desk. But all the solutions need to be looked at. There are too many instances of unconventional solutions turning out to be the perfect solution for a problem. While it may be a trite and overused term, 'thinking outside the box' is the best description of what should happen here.

Winnnow the list of possible solutions objectively

That's not to say that standard solutions should be tossed aside either. *All* solutions, both conventional and unconventional need to be looked at with the same objectivity. Don't worry that you will have such a large list of solutions that you won't be able to pick a solution. There are always going to be requirements that will help determine the solution. For example, while a fibre - channel SAN may be a faster way to serve files, if you don't have a fibre-channel setup in place, the fiscal and time costs of such a solution may remove it from the list.

Space limitations are an example of a factor that is going to apply to any solution, and is platform neutral. If you only have a small amount of space in your server room for a new server, then a solution that involves hundred - station Linux clusters may not be practical at this time. Network limitations are another example. That reconditioned AS/400 may indeed be a cheap and fast server, but if it can only use Twinax and you only can implement Ethernet, then this is not a good solution.

The point is, make sure that you use unavoidable limitations to winnow your solutions list first. Network, physical, fiscal, these are all limits that should take precedence over the operating system and platform for the server.

Standards are good, but in moderation

So now you have a manageable solutions list. What about computing standards? Well, as long as you don't go overboard, and apply standards to where they need to be applied, they can be an aid. Too many companies standardize on operating system or application, when they would be better off standardizing on data format. If you want to ensure uniformity of final output, standardizing on Windows and Word or Solaris and LaTeX will not do nearly as much good as standardizing your fonts, image formats, and data formats. Limiting your font usage, standardizing on a small number of image formats, such as TIFF, JPEG, MPEG, EPS, etc., and using Acrobat as your final output of choice is going to give you all the benefits of standardization, but will leave you with a far more capable tool box than a single platform and application will. It also means that if the preferred application or platform is under attack, you can still get work done.

This does not mean just randomly seed different platforms about your network. First, if you have 100 people in a group using Windows, and one person using a Mac, all you create are problems for yourself. Implementing a different platform has to

Felt Tip Software presents

Sound Studio 2.0

Built for Mac OS X

With Sound Studio 2.0, you can:

Digitize cassette tapes and vinyl records to make CDs

Record live performances and interviews

Record your favorite radio programs with Timer Recording

Edit out dead air, glitches, and mistakes in recording

Apply any of 19 effects including Reverb and Chorus

Create new sounds and loops

Sound Studio is an audio recording and editing application built for Mac OS X. It can also run on Mac OS versions 8.1 and up. Some systems may need a USB audio input device to record.

Only \$49.99

Download 14-day trial version at

<http://www.felttip.com/ss/>



felt tip software

807 Keely Place, Philadelphia, PA 19128, USA
www.felttip.com

<http://www.scientific.com>

Professional Software Developers

Looking for career opportunities?

Check out our website!

Nationwide Service

Employment Assistance

Resume Help

Marketability Assessment

Never a fee

Scientific Placement, Inc.

800-231-5920 800-757-9003 (Fax)

das@scientific.com

be done in a planned logical way. Willy - nilly is a fun way to do many things, but network design is not one of them.

Common knowledge is always wrong

One of the signs that a solution is going to be bad is when it starts with any variant of "Well, everyone *knows*..." There is nothing in computing that everyone knows that is ever universally correct. "Everyone knows" that AppleTalk is chatty and getting rid of it will make things better. Well, to quote an Apple engineer on the AppleShareIP team:

"On Bandwidth:

An idle connection to an AppleShare server (via IP) sends 2 tickle packets of about 64 bytes in size every 30 seconds (call it 4 bytes/second or 0.00024% of a 10 Mega Bit [10 Base-T] connection <I may be off by a factor of 10 either way, its early>). When transferring files, AFP is just as efficient as any other well implemented IP protocol, a single client can, under ideal conditions fill the pipe with minimal overhead. For a 16k read/write we have 28 bytes of AFP/DSI protocol info on top of 780 bytes of TCP/IP protocol info for a payload efficiency of about 91% (it takes 12 packets to move 16k of data).

So maybe everybody knows nothing. The point here is don't assume anything. The solution is good or bad based on the facts, not assumptions, attitudes, personal prejudices, the magnetic pull of the Moon, Kentucky windage, etc. If you let anything but the facts and reality guide your selection of a solution, then you may as well throw darts at the list, you'll have as much luck that way as any, and it's probably more fun.

IMPLEMENTING DIVERSITY

So, the solution to the new file server is a new server. It has to be able to authenticate against your active directory servers transparently, it has to support Windows clients smoothly, and it has to fit into your Veritas backup solution. Congratulations, you have a multitude of platforms to pick from. Samba can easily handle Windows clients and use an upstream Active Directory server to handle authentication requests. Veritas supports a wide range of client platforms, including Mac OS X, so you can freely pick from any of them.

You'll get a new, faster file server. Your users will be happier because they get to their files faster, with the same reliability. Because you can choose from a wider range of vendors, you get better competition for your business, which means your upfront costs are smaller. But even better, what happens when a Windows virus comes ripping through your network and hits that Linux file server?

It dies. Stops. That machine isn't infected. Those files are safe. What happens when some cracker larvae is using the latest Solaris hack to get past your firewall, and hits your Mac OS X file server? The same thing. He now has to figure out what this operating system is, and then find ways to crack it. He may indeed find a new crack, but that Solaris - specific hole that he used is closed at least here.

By having a genetically diverse network, you aren't losing anything, except the loss of security and capability that comes with a building full of genetically identical clones. Your up front costs don't have to be any higher. You may have a learning curve on the new platform, but those aren't as bad as they seem

to be, and the Internet has terabytes of information that will help you along. By having a mixture of server platforms and client platforms you create firebreaks on your network. You ensure that no single attack that is targeted at a single vulnerability is able to completely compromise your entire network unchecked. While Code Red may load down an iPlanet web server, it's certainly not going to abuse it in the same way as it will an unpatched IIS server. Outlook viruses become merely amusing if you aren't using Outlook as your only email client.

In addition, you gain a whole host of capabilities that you simply cannot achieve in a homogenous environment. Unix, Windows, Mac OS X, AS/400s, NetWare, *BSD, Linux, et al all have unique strengths and weaknesses that can compliment each other. There are products that only exist on a single, or small number of platforms that can be of great use to you, but only if you have that platform available.

Even better, when you combine a genetically diverse network with a well - thought out set of prophylactic measures, such as antivirus programs and intrusion monitors, both methods become more effective and secure. Even if one of your low - infection platforms does get infected, the damage that occurs will be limited because the other platforms won't be infected. Not only is the damage mitigated, but you also have more time to prevent a similar problem on the other platforms. The prophylactic protections don't take up as much of your time, because they have a genetic backup in place. They have less to watch out for, because your firebreaks are intercepting and halting much of the damage before it hits vulnerable systems. As well, other prophylactic measures can help you stop the problem before it hits vulnerable or targeted systems.

Genetic diversity is just another, less common way of removing a single point of failure. If you aren't going to do that, then why bother with RAID and failover servers, etc?

CONCLUSION

Genetic diversity isn't just some fad, or some keen idea that has no basis in reality. In fact, it has millennia of proof that it is a good tactic. It is a proven way to keep any population healthy and functional, no matter if you are talking about potatoes, humans, trees, cows, military forces, or computers. It may take more work than a clone farm, but the benefits are real, tangible, and undeniable. It's not a panacea, but it can, and will make your network stronger and more capable.

In the end, there is no magic bullet. Every form of protection, including genetic diversity on your network has a weakness. You have to combine network genetics and prophylactic measures, along with a lot of planning, to achieve the best results. Use both. The next time you buy a new box, if you already have a lot of that platform, see if maybe you can get the same, or even better results from a different platform. You'll learn more, you'll gain more capabilities, and the next time the Legion Of Bad People unleashes some hideous Windows, or Linux virus, you'll have a much better time than your counterparts in the lands of Windows and Linux clones.

Introducing 360 One VR



Developed by EyeSee360 and Kaidan, 360 One VR™ is an amazing product that captures a complete 360° panoramic image in a single camera shot. Years of extensive research have produced an innovative solution that creates an immersive image without the restrictions or compromises normally associated with panoramic photography.

360 One VR consists of a lightweight and rugged proprietary optical device and the innovative EyeSee360 PhotoWarp™ software. The unique mirrored optic provides a complete 360° horizontal panorama with an outstanding 100° vertical field-of-view (50° above and 50° below the horizon).

This vertical field-of-view is substantially greater than competitive offerings and brings single-shot panoramic technology into the mainstream. With 360 One VR, single-shot panoramas can be used for interior shots and other situations where it is important to look up and down as well as side-to-side.

When coupled with a 3 megapixel or greater digital camera, the 360 One VR, with its precision coated glass optics, provides a level of quality that rivals conventional "stitched" solutions requiring two or more images. Since 360 One VR requires only a single shot to capture the full 360° view, action scenes with moving objects are easily photographed.

360 One VR can capture detailed panoramic images in crowd scenes, at concerts, on city streets, at sporting events, and even from moving vehicles. Photographers avoid the stitching or seaming required by competitive multiple-shot systems, thereby eliminating the possibility of visible seams or exposure differences across the panorama.

The EyeSee360 PhotoWarp™ software quickly and easily processes a 360 One VR photograph into a viewable panoramic image. The straightforward, simple-to-use interface automatically generates QuickTime VR® panoramas, thumbnail images and web pages. PhotoWarp can also produce unwarped images suitable for Java™-based panoramic viewers and printed output.

Visit our website, <http://www.360OneVR.com> for more info.



Macworld
**BEST
OF SHOW**
2002

KAIDAN

Kaidan Incorporated
703 East Pennsylvania Blvd. Feasterville, PA 19053, USA

Phone: 215-364-1778 • Fax: 215-322-4186 • info@kaidan.com • <http://www.kaidan.com>



<http://www.eyesee360.com>

Kaidan is a trademark of Kaidan Incorporated. EyeSee360 and PhotoWarp are registered trademarks of EyeSee360 Inc. 360 One VR is a trademark of EyeSee360 and Kaidan Incorporated. QuickTime and Mac OS X are registered trademarks of Apple Computer, Inc. Windows is a registered trademark of Microsoft Corporation. Coolpix is a registered trademark of Nikon, Inc. Java is a trademark of Sun Microsystems. All other trademarks are the property of their respective owners. Specifications and equipment are subject to change without any notice or obligation on the part of the manufacturer. All panoramic images in this brochure were shot with the 360 One VR. © 2002 Kaidan Incorporated

By Dan Wood, Alameda CA

The Beauty of Categories

Use This Objective-C Feature to Make Your Cocoa Code Cleaner

Ask any experienced Cocoa programmer what they like the most about Objective-C and the answer will invariably be "categories." Categories is one of the features of Objective-C, not found in Java or C++, that raises the body temperature of developers if you suggest they use another language.

What is It, and Why Use It?

A category is an extension of an existing class. But unlike inheritance, in which you create a new class that descends from another class, a category is like a remora, attaching itself to the belly of a shark and getting a free ride. By creating a category, you add new methods to an existing class, without needing to create a new one.

Writing in a language without categories, the programmer is often faced with the need to perform minor operations, acting upon an object for which source code is unavailable. These routines might end up as methods in the application class that needs to perform those operations, although that doesn't promote reuseability, since the operations are tied in with the enclosing class. A better approach, one more commonly used, is to collect these operations into a utility class.

On an open-source web application framework that I worked on, called Janx (available at www.bearriver.com) there is a string utilities class, for example. This class has operations to parse strings representing dollars and cents, encode a string for HTML display, generate a hexadecimal representation, build an MD5 digest from a string, and so forth. Each of these methods takes a string to operate upon as one of its parameters.

This "utility class" approach isn't particularly elegant either. Dissimilar operations tend to be grouped together into the same class. Each method must be passed in the object to operated upon as a parameter, which means that the functions that you write look and operate differently from methods that are part of the class, even if they perform similar operations.

Another approach to extending functionality is to create a subclass of an existing framework object, and add your new

functionality into the subclass. For instance, you might subclass an existing "image" class to add operations. The problem is that you must now be sure to work only with instances of your new class; any objects that aren't must be converted.

If you are programming in a language such as C++ or Java without categories, though, you just deal with these limitations; they may not seem like limitations at all.

When you write an application in Cocoa using Objective C, you have the ability to put such functions directly into an existing class by creating a category on that class. No, you don't recompile the class with new methods in the file; in fact you usually don't have the source code to the class you are adding to.

UTILITIES VS. CATEGORIES

Let's take a look at how this might be done by implementing a utility function to strip quote marks off of a string. (We'll implement them both in Objective-C just to keep the playing field level.) We implement it as a method in a string utility class in **listing 1** and **2**; we implement it as a category on NSString in **listing 3** and **4**.

Listing 1: StringUtilities.h

```
#import <Foundation/Foundation.h>
@interface StringUtilities
+ (NSString *) stripQuotes:(NSString *)inString;
@end
```

Listing 2: StringUtilities.m

```
#import "StringUtilities.h"
@implementation StringUtilities

+ (NSString *) stripQuotes:(NSString *)inString
{
    NSString *result = inString; // Return inString if no stripping needed
    int len = [inString length];
    if (len >= 2
        && '"' == [inString characterAtIndex:0]
        && '"' == [inString characterAtIndex:len-1])
    {
        // Get the substring that doesn't include first and last character
        result =
            [inString substringWithRange:NSMakeRange(1,len-2)];
    }
}
```

Dan Wood wrote Watson for Mac OS X, a Cocoa application that connects to a variety of Web services. You can reach him at dwood@karelia.com.

```

}
return result;
}

```

Listing 3: NSString+misc.h

```

#import <Foundation/Foundation.h>
@interface NSString ( misc )
- (NSString *) stripQuotes;
@end

```

Listing 4: NSString+misc.m

```

#import "NSString+misc.h"
@implementation NSString ( misc )

- (NSString *) stripQuotes
{
    NSString *result = self; // Return self if no stripping needed
    int len = [self length];
    if (len >= 2)
        && ' ' == [self characterAtIndex:0]
        && ' ' == [self characterAtIndex:len-1])
    {
        // Get the substring that doesn't include first and last character
        result = [self substringWithRange:NSMakeRange(1,len-2)];
    }
    return result;
}

```

The implementations of these category looks much like the utility class; the main difference is that the string to operate upon is not passed in as a parameter; it is accessed with the self keyword. Things start to look different when you compare code that uses the category instead of a utility class. Here are snippets that use each approach.

Snippet using a utility class

```

NSString *stripped =
[StringUtilities stripQuotes:theValue];
[lineDict setObject:stripped
 forKey:[theValue uppercaseString]];

```

Snippet using a category

```

NSString *stripped =
[theValue stripQuotes];
[lineDict setObject:stripped
 forKey:[theValue uppercaseString]];

```

The code using the category is quite a bit cleaner because we don't have to be conscious of a separate utility class; it is just another operation on the string, just like the built-in uppercaseString method on the last line.

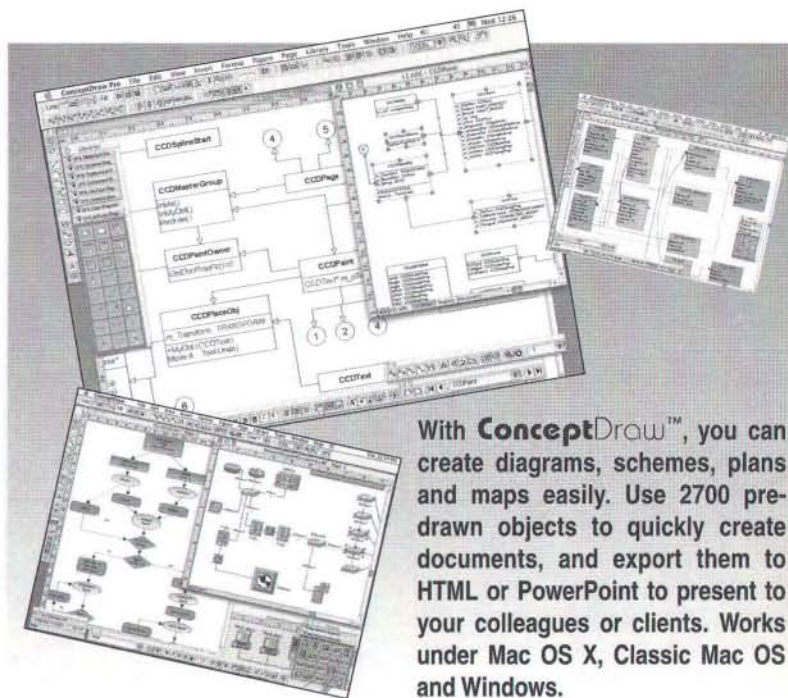
WRITING CATEGORIES

A category must have an @interface and @implementation section, just as a class. After the name of the class being added to is an arbitrary name which describes what the category is for, in parentheses. The example above uses "misc" as its name.

Normally, a category on a class gets its own ".h" and ".m" file; a convention is to name the file based on the class name concatenated with "+" to the category name. For example, the file NSString+bitmap.m would be expected to hold @implementation NSString (bitmap). This is not strictly

ConceptDraw™ PROFESSIONAL

Cross-platform diagramming and drawing tool



With ConceptDraw™, you can create diagrams, schemes, plans and maps easily. Use 2700 pre-drawn objects to quickly create documents, and export them to HTML or PowerPoint to present to your colleagues or clients. Works under Mac OS X, Classic Mac OS and Windows.

Use ConceptDraw Pro for:

- UML
- Network Diagrams
- Flowcharts
- Web Site Schemes
- Interface Modelling
- Process Flow



neccesary; however, you could make a quick category interface and implementation right in your class file that makes use of the category; this would only be practical if it was not needed outside of the associated class.

Methods are declared and implemented just as they would be for any standard Objective-C's methods. Keep in mind, however that self is the class that you are implementing; feel free to send messages to self to operate on that object.

The one big limitation on categories is that you can only add functionality; you cannot add new data members to the class. There are no curly braces in the @interface section of a category. If you feel the need to add data members, you may want to consider subclassing instead.

USING CATEGORIES

The best thing about categories is that you can add whatever features to Cocoa you'd like to that you feel are "missing." Frustrated that `UIImage` lacks the `+[UIImage imageData:]` method? Add it in yourself! You can write generic categories and use them on all your projects, and make use of them as if you were using functionality of the classes provided by Apple. Or, you can create categories on an object as needed, whenever it seems more intuitive to extend the functionality of a Cocoa class rather than write a function to act upon that object.

You can even use categories on your own code, to help factor your application's classes into smaller, more manageable chunks. For instance, you might create separate categories to partition your document controller into preferences management, window management, and general functionality. Doing so makes your files smaller and makes your project more navigable. Cocoa itself makes heavy use of categories in this manner; it allows classes to be created in one library (such as Foundation Kit) and then extended in another (such as Application Kit).

One of the best places to use a category is to split up your class's private methods from its public ones, to overcome a limitation in Objective-C. Unlike C++ and Java, there's no way to specify the access of a method using keywords. So the solution is to create a new @interface for your category at the top of your class's ".m" file, holding the methods you do not want to be exposed in the ".h" file. This category would have a name such as "private" to indicate its purpose. Below that, the @implementation section of your class can then hold the implementation of both the public methods (declared in the ".h" file) and the private methods (declared in your private category). Other classes will not be able to see your private methods.

Usually, you will find yourself adding categories to classes in the Foundation Kit, because this kit tends to hold containers and utilities. You can even add categories to `NSObject` so that any object can respond to your new functionality. When there is a technique that requires bridging into Carbon or Core Foundation to accomplish your task, you could wrap it into a

category on a related class (or even find one online that somebody else has already written), so that if such functionality were to make its way into a future version of Cocoa, your code wouldn't have to change much.

EXAMPLES

Where you make use of categories is limited only by your imagination. It is useful to look at other people's source code just to get a sense of what kinds of categories are possible. Many source code packages are available for downloading at softtrak.stepwise.com.

Here are a few examples that I have used in my own code. To make use of these, you would need to create @interface and @implementation sections following the guidelines above.

CATEGORY FOR UIImage

A method to set an image size to be the size of its associated `NSBitmapImageRepresentation` so that the image displays at full size of 72 DPI. It finds the first bitmap it can, and sets the size of the bitmap and of the image to the pixel width and height.

```
- (UIImage *) normalizeSize
{
    NSBitmapImageRep *theBitmap = nil;
    NSArray *reps = [self representations];
    NSSize newSize;
    int i;

    for (i = 0 ; i < [reps count] ; i++)
    {
        NSImageRep *theRep = [reps objectAtIndex:i];
        if ([theRep isKindOfClass:[NSBitmapImageRep class]])
        {
            theBitmap = (NSBitmapImageRep *)theRep;
            break;
        }
    }
    if (nil != theBitmap) // Found a bitmap to resize
    {
        newSize.width = [theBitmap pixelsWide];
        newSize.height = [theBitmap pixelsHigh];
        [theBitmap setSize:newSize]; // resize bitmap
        [self setSize:newSize]; // resize image
    }
    return self;
}
```

Category for NSBundle, NSDictionary, NSString, etc.

A comparison method (passing in another object of the same) so that you can sort an array of those objects by some property, using `-[NSMutableArray sortUsingSelector:]`. For example, you could sort an array of dictionaries by the value of their "name" key by passing in the selector for the following method.

```
- (NSComparisonResult) compareSymbolName:
    (NSDictionary *) inDict
{
    NSString *myName = [self objectForKey:@"name"];
    NSString *otherName = [inDict objectForKey:@"name"];
    return [myName caseInsensitiveCompare:otherName];
}
```

Your Data Isn't an Important Part of the Job — It *IS* the Job.

VXA® FireWire

The High-Performance Tape Storage Solution For Apple Users

*"I can stick
my entire
hard drive
onto a tape
7 times
—that's just
plain cool!"*

Other backup media, such as CD-RWs and DVD-RAMs, simply can't compare to VXA-I tape technology when it comes to capacity, transfer rate and overall price.

Fast, economical and easy-to-use, VXA FireWire offers:

Ultimate File Security

- 100% Data Restore and Interchange
- Plug-and-Play Connectivity

Sharable Media

- Multi-Gigabyte File Transport
- Portable, Hot-Pluggable

Real Time Digital Video

Cross-Platform Compatibility

- FireWire/IEEE1394/iLink Supported By All Major Manufacturers



Technology	Capacity in MBs	Transfer Rate in MB/sec	Price per MB
VXA Tape	33000	3	\$0.03
Imation Travan	10000	1	\$0.05
DVD RAM	9400	2.7	\$0.05
CD RW	700	1.9	\$0.50
Zip	250	1.2	\$0.76

LOWEST
COST per MB
OF ANY
Technology!

Call Exabyte sales at 1-800-774-7172
or visit us on the web at www.exabyte.com



Tape Storage By
Exabyte®



Category for NSString

A method to return an attributed string as a blue underlined hyperlink, so that text fields can respond to link clicks as in a web browser. Text in an NSTextView with these attributes will send the message of textView: clickedOnLink: atIndex: to the view's delegate.

```
(NSAttributedString *)hyperlink
{
    NSDictionary *attributes=
    [NSDictionary dictionaryWithObjectsAndKeys:
     [NSNumber numberWithInt:NSSingleUnderlineStyle],
     NSUnderlineStyleAttributeName,
     self, NSLinkAttributeName, // link to the string itself
     [NSFont systemFontOfSize:[NSFont smallSystemFontSize]],
     NSFontAttributeName,
     [NSColor blueColor], NSForegroundColorAttributeName,
     nil];
    NSAttributedString *result=
    [[[NSAttributedString alloc]
     initWithString:self
     attributes:attributes] autorelease];
    return result;
}
```

Category for NSWorkspace

A method to return the path of the current user's temporary directory. This makes use of the Carbon FindFolder() API, and then converts the C string into an NSString.

```
(NSString *) temporaryDirectory
{
    char s[1024];
    FSSpec spec;
    FSRef ref;
    short vRefNum;
    long dirID;

    if ( FindFolder(
        kOnAppropriateDisk, kChewableItemsFolderType, true,
        &vRefNum, &dirID ) == noErr )
    {
        FSMakeFSSpec( vRefNum, dirID, "", &spec );
        if ( FSpMakeFSRef(&spec, &ref) == noErr )
        {
            FSRefMakePath(&ref, s, sizeof(s));
            return [NSString stringWithCString:s];
        }
    }
    return nil;
}
```

Category for NSSet, NSArray, etc.

A method to build a string listing the strings in a collection, separated by commas. It enumerates through all objects in the structure, adding each string and then adding a comma. It then removes the extra comma (and space) at the end, after the list is traversed.

```
(NSString *) show
{
    NSString *result = @""; // empty string if none in
    collection
    NSMutableString *buffer = [NSMutableString string];
   NSEnumerator *theEnum = [self objectEnumerator];
    NSString *theIdentifier;

    while (nil != (theIdentifier = [theEnum nextObject]))
    {
        [buffer appendString:theIdentifier];
        [buffer appendString:@", "];
    }
    // Delete final comma+space from the string
    if (![buffer isEqualToString:@""])
    {
        [buffer deleteCharactersInRange:NSMakeRange(
            [buffer length]-2, 2)];
        result = [NSString stringWithString:buffer];
    }
    return result;
}
```

CONCLUSION

Hopefully you have been convinced that categories are a useful construct for programming in Cocoa. If you're not using Objective-C, you can certainly function without them. But if you are, then categories are a great way to make your code more readable, more reuseable, more maintainable, and simpler.

Impact your bottom line while working with dedicated proponents of Apple technologies

With us, your only constant is success. Our custom solutions help our clients excel their business objectives and effectively target their audience. That's why we suggest that we run your technology while you play the winning shot.

Core Competencies

- WebObjects development
- Mac OS X application development
- Cocoa application porting
- Carbon porting
- Mac OS product maintenance
- Windows/Mac OS/Unix porting
- Cross-platform development

Service Offerings

- Offshore Project Development
- On-Site Staff Augmentation
- Off-Site Project Development
- User Training

For More Information

visit <http://www.avestacs.com>
Email: mithu@avestacs.com



Avesta Computer Services, Ltd.
USA . EU . Asia



by Bob Boonstra, Westford, MA

MATCHSTICKS

Some time ago, Robin Landsbert sent me a suggestion for a Challenge based on a game he called Nim. In Robin's version of the game, matchsticks were arranged in rows forming a triangle, one matchstick in the top row, two in the next, three in the next, etc. Two players take turns removing one or more matchsticks from any single row of the board. The object is to make your opponent take the last matchstick.

A little research suggests that this version of the game might not be very difficult. So, in the tradition of the Challenge, we will add a few twists that might make the game (and the Challenge) more interesting. First, we will arrange our matchsticks in a square grid instead of a triangular one, and allow players to remove matchsticks from either a single row or a single column on a given turn. Second, we will not put a matchstick in every position in the grid, leaving a small number of positions empty, perhaps on the order of 10%. Third, we will restrict a player's moves to removing matchsticks with no intervening holes. That is, a player can remove the $n+1$ matchsticks in row r located in columns c through $c+n$ only if a matchstick is present in each of those locations. And finally, we will play two versions of the game, one where the player taking the last matchstick loses the game, and one where the player taking the last one wins the game.

The prototype for the code you should write is:

```
void InitMatchsticks(
    short dimension,
    /* game is played on a square board of size dimension x dimension */
    const char *board,
    /* board[row*dimension + col] is board cell (row,col) */
    /* board[][]==1 represents a matchstick, ==0 represents an empty cell */
    bool playFirst,
    /* true if you will play first in this game */
    bool lastMatchstickLoses,
    /* true if taking the last matchstick loses the game,
       false if taking the last one wins the game */
    short opponent,
    /* identifier for your opponent in this game */
);

void OpponentMove(
    bool playingRow,
    /* true if opponent played along a row, false if along a column */
    short rowOrColumnNumber,
    /* number of the (origin zero) row (playingRow==true) or
       column (playingRow==false)
       that the opponent played */
    short startingColOrRow,
    short endingColOrRow,
    /* if playingRow==true, the opponent played from
       (row,col)==(rowOrColumnNumber,startingColOrRow)
       to (row,col)==(rowOrColumnNumber,endingColOrRow)
       if playingRow==false, the opponent played from
       (row,col)==(startingColOrRow,rowOrColumnNumber)
       to (row,col)==(endingColOrRow,rowOrColumnNumber)
    */
    const char *board,
    /* board after your opponent's move */
);

const char *YourMove(
    bool *playingRow,
    /* true if you played along a row, false if along a column */
```

```
    short *rowOrColumnNumber,
    /* number of the (origin zero) row (playingRow==true) or
       column (playingRow==false)
       that you played */
    short *startingColOrRow,
    short *endingColOrRow
    /* if *playingRow==true, you played from
       (row,col)==(*rowOrColumnNumber,*startingColOrRow)
       to (row,col)==(*rowOrColumnNumber,*endingColOrRow)
       if *playingRow==false, you played from
       (row,col)==(*startingColOrRow,*rowOrColumnNumber)
       to (row,col)==(*endingColOrRow,*rowOrColumnNumber)
    */
    /* return value is a pointer to a board after your move */
);
```

The objective of the Challenge is to win as many games as possible against your fellow contestants, while expending as little execution time as possible. Each game begins with a call to your InitMatchsticks routine, where you are given the dimension of the game, the initial board configuration, the identity of your opponent, whether or not you playFirst, and whether the objective is to take or not take the last matchstick (lastMatchstickLoses). When it is your turn to move, your YourMove routine describes the move you are making (playingRow, rowOrColumnNumber, startingColOrRow, endingColOrRow) and returns a pointer to your view of what the board looks like after your move. When your opponent moves, your OpponentMove routine is provided with a description of the opponent's move, and the board configuration after that move.

The Challenge will be scored as a round robin tournament, or another fair scheduling mechanism. Each player will play first and play second against each scheduled opponent an equal number of times for each test case. Each player will play to win by taking the last matchstick, and play to win by making the opponent take the last matchstick, an equal number of times against each scheduled opponent for each test case. A player's score will be dimension^2 points for each win, minus a penalty of 10 points per millisecond of execution time. You can earn a bonus of up to 25% of your score based on a subjective evaluation of the clarity of your code and commentary.

This will be a native PowerPC Carbon Challenge, using the Metrowerks CodeWarrior Pro 7.0 development environment. Please be certain that your code is carbonized, as I may evaluate this Challenge using Mac OS X. Unfortunately, this Challenge cannot accommodate alternative development environments, because pairs of solutions need to compete against one another in a single executable.

WINNER OF THE MARCH, 2002 CHALLENGE

The March Challenge required contestants to solve the Megaminx, a twelve-sided puzzle in the shape of a dodecahedron. Each of the twelve faces of the Megaminx can be rotated clockwise or counter-clockwise, with five consecutive rotations of a face in the same direction bringing the face back to its original position. Each face is divided into eleven facelets, five corner facelets that each

**Bridge the communication gap,
make the right connection —
with innovative solutions at**

**DEV
DEPOTSM**

www.devdepot.com

**Griffin Pro Speaker
Breakout Cable**

\$23⁹⁵

Connect standard speakers to your
Mac pro speaker socket!



**SMARTLINK USB
File Transfer Cable**

\$39⁹⁵

Transfer files over USB, between
Mac, PC's, or across platforms!



A silver braided cable with a DVI connector on one end and a 15-pin ADC connector on the other.

DVIator DVI to ADC Adapter

Connect flat panel monitors to your DVI video source!
(yes we have PCI DVI video cards too!)

\$147⁹⁵

A black USB-to-video adapter cable with a USB connector on one end and composite video (red, white, and yellow) connectors on the other.

Belkin USB VideoBus II for Mac

Fast and easy video capture over USB!

\$79⁹⁵

A small, round, silver USB-to-audio adapter with a USB connector on one side and 3.5mm audio jacks on the other.

Griffin iMic

Sound in and out over USB!

\$32⁹⁵

**Don't have the right port?
Maybe you just need the right cable!**

border three faces, five edge facelets that each border two faces, and one center facelet. The faces are colored with six colors, opposite faces sharing the same color. The input for the Challenge was a sequence of files that each described a scrambled Megaminx, and the required output was a sequence of rotations that solved the puzzle. Scoring was based on the execution time required to solve the scrambled puzzles. Contestants earned up to a 25% reduction in their time if they also displayed the puzzle solution.

Two contestants, Ernst Munter and Allen Stenger, submitted solutions for this Challenge. Both contestants acknowledge the information provided at two Megaminx web sites, one provided by Meffert's Puzzles at <http://www.meffertspuzzles.com/puzzles/megasol1.html>, and another by W. D. Joyner. Ernst used the approach described in <http://web.usna.navy.mil/~wdj/megam.htm>, one that solved the problem quickly, but generated solutions with a large number of moves. Ernst first moved the corner pieces to the proper positions, then moved the edge pieces to the proper positions, then oriented the corners, then oriented the edges. Allen took the nine-step approach described at the Meffert site, augmented with a modification from <http://web.usna.navy.mil/~wdj/megaminx.htm>, an approach that generated shorter move sequences, but took more execution time.

Both contestants provided display options in their entries. Ernst's program has a compile-time option to generate a two-dimensional depiction of the Megaminx as the solution is generated. He included an option to display macro moves in a single step, which made it easier to see what was going on. Allen's entry has a separate program, written in Cocoa and using OpenGL to display a three-dimensional Megaminx. Allen included options to read in a puzzle description file and a sequence of moves, controls to rotate the viewpoint, and controls to rotate a slice of the puzzle.

By the stated rules of this contest, the solution requiring the least amount of execution time, after considering the display bonus, is the winner. Congratulations to **Ernst Munter** (Kanata, ON, Canada) for winning the Megaminx Challenge. I am taking the somewhat unusual step, however, of providing both solutions in the online archive, and printing the better-commented solution by **Allen Stenger** in this article.

The table below lists, for each of the solutions submitted, the total execution time in microseconds, the time reduction awarded for providing a display option, the net penalty points after subtracting the bonus from the execution time, and the cumulative number of moves required to solve the ten test cases used to evaluate solutions. It also lists the programming language of each entry. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

Name	Exec. Time (microsecs)	Display Bonus	Penalty Points	Moves	Language
Ernst Munter (275)	37331	25%	27998	15030	C++
Allen Stenger (39)	347335	25%	260501	6440	C++/ObjC

TOP CONTESTANTS ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points (24 mo)	Wins (24 mo)	Total Points
1.	Munter, Ernst	275	10	862
2.	Saxton, Tom	52	1	210
3.	Stenger, Allen	49	1	114
4.	Rieken, Willeke	46	2	134
5.	Wihlborg, Claes	40	2	49
6.	Taylor, Jonathan	37	1	63
9.	Gregg, Xan	20	1	140
10.	Mallett, Jeff	20	1	114
11.	Cooper, Tony	20	1	20

... AND THE TOP CONTESTANTS LOOKING FOR A RECENT WIN

In order to give some recognition to other participants in the Challenge, we also list the high scores for contestants who have accumulated points without taking first place in a Challenge during the past two years. Listed here are all of those contestants who have accumulated 6 or more points during the past two years.

Rank	Name	Points (24 mo)	Total Points
7.	Sadetsky, Gregory	22	24
8.	Boring, Randy	21	144
13.	Schotsman, Jan	16	16
14.	Shearer, Rob	15	62
15.	Hart, Alan	14	39
16.	Nepsund, Ronald	10	57
17.	Day, Mark	10	30
18.	Desch, Noah	10	10
19.	Flowers, Sue	10	10
20.	Maurer, Sebastian	7	108
21.	Leshner, Will	7	7
22.	Miller, Mike	7	7

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points
2nd place	10 points
3rd place	7 points
4th place	4 points
5th place	2 points
finding bug	2 points
suggesting Challenge	2 points

Here is Allen's Megaminx solution:

CSolver.cpp

```
////////////////////////////////////////
//
// Megaminx (MacTech Programmer's Challenge, March 2002)
// Written by Allen Stenger, March 2002
//
// Conceptually we rotate colors rather than faces; this simplifies the problem of
// determining the orientation of each edge and corner piece.
//
// We follow the solution given by Meffert's Puzzles and Novelties;
// see http://www.mefferts-puzzles.com/puzzles/megasol1.html
//
// Terminology: corner piece is at a vertex of the Megaminx and has three facelets,
// edge piece is at an edge between two faces and has two facelets. The smaller
// pentagons in the center of each face never move away from the face and so we
// ignore them.
//
// A vertex can be specified by its vertex number, however edges don't have numbers
// and are usually specified by the two faces they lie on. There is a variety of constant
// tables for walking through the faces.
//
// COLOR AMBIGUITY
//
// Because the same colors are used for two faces, it appears that there might be some
// ambiguity in the pieces; that is, radially-opposite corners have the same colors, and
// and radially-opposite edges have the same colors, so how do we know whether to
// place a corner or edge in the Northern or Southern part of the Megaminx?
//
// *The corners are actually not ambiguous because the orientations
// are different; so for example there are two corners with
// colors 1,2,3, but the one on the North Pole has the colors
// 1,2,3 in clockwise order and the one on the South Pole has
// 1,2,3 in counter-clockwise order. Therefore we can always
// tell from the corner which part of the Megaminx it goes in.
// *The edges really are ambiguous. It is not necessary to put
// each edge back in its original place, but in some situations
// we would get to Step 8 and be unable to orient the South
// Pole edges because of an earlier placement we made. To solve
// the Megaminx we must follow some parity rules; see
//
// Coreyanne Rickwalt, "The Fundamental Theorem of the
// Megaminx", http://web.usna.navy.mil/~wdj/megaminx.htm.
//
// We will detect the problem case in Step 6 and take evasive action.
//
// A simple example of the problem is a Megaminx that is solved except
// for the two edges:
// 8,7,3
// 9,7,2
// This one cannot be solved by the published Meffert method because
// the South Pole edges are not correctly placed in Step 8.
//
////////////////////////////////////////

#include "CSolver.h"
#include "CMegaminx.h"
#include "CMegaminxApp.h"

#include <cassert>
#include <sstream>

// some fixed faces we use
const int kSouthPoleFace = 7;

////////////////////////////////////////

CSolver::CSolver(CMegaminx& rMega) :
fMega(rMega)
{
}

CSolver::~CSolver()
{
}

void CSolver::Solve()
{
    // call all the solution steps
```

```
Step3():
Step4():
Step5():
Step6():
Step7():
Step8():
Step9():
Step10():
Step11():
}

void CSolver::DoLUU(CMegaminx::face_t leftFace,
CMegaminx::face_t rightFace)
{
    fMega.WriteComment("DoLUU");
    fMega.Slice(leftFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCounterCW, 1);
}

void CSolver::DoRUU(CMegaminx::face_t leftFace,
CMegaminx::face_t rightFace)
{
    fMega.WriteComment("DoRUU");
    fMega.Slice(rightFace, CMegaminx::eCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCW, 1);
}

void CSolver::DoRLL(CMegaminx::face_t leftFace,
CMegaminx::face_t rightFace)
{
    fMega.WriteComment("DoRLL");
    fMega.Slice(rightFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCounterCW, 1);
}
```

high quality - competitive rates - 16 years experience - award winning

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

Full Spectrum Software

Development & Testing

Device Drivers

Porting

Plug-ins

TCP/IP

Carbon / OSX

Cross Platform Development

One Bridge Street
Newton, MA 02458
617-965-0029
www.FullSpectrumSoftware.com

competitive rates - 16 years experience - award winning - high quality

```

void CSolver::DoLLL(CMegaminx::face_t leftFace,
CMegaminx::face_t rightFace)
{
    fMega.WriteComment("DoLLL");
    fMega.Slice(leftFace, CMegaminx::eCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(leftFace, CMegaminx::eCounterCW, 1);
    fMega.Slice(rightFace, CMegaminx::eCW, 1);
}

void CSolver::VisitAllCorners(CCornerVisitor &aVisitor)
{
    for (int i = 0; i < CMegaminx::kNumVertices; i++)
        aVisitor.VisitCorner(i);
}

bool CSolver::CheckEdgeParity()
{
    // this holds the permutation of the South edges. It is
    // in two 5-edge pieces:
    // 0-4: South Equator edges, indexed same as SouthEqEdge arrays
    // 5-9: South Pole edges, indexed same as SouthPoleEdge arrays + 5
    // The entries are also these indices; perm[i] contains the edge
    // index that edge i will go to when the Megaminx becomes solved.
    // Therefore the entries in perm are the numbers 0-9 in some
    // permuted order.
    int perm[10];

    for (int i = 0; i < 10; i++)
        perm[i] = 0;

    // South Equator edges
    for (int i = 0; i < CMegaminx::kNumSouthEqEdges; i++)
    {
        CMegaminx::color_t c0 =
            fMega.EdgeFaceletColor(fMega.kSouthEqEdgeL[i],
                                   fMega.kSouthEqEdgeR[i]);
        CMegaminx::color_t c1 =
            fMega.EdgeFaceletColor(fMega.kSouthEqEdgeR[i],
                                   fMega.kSouthEqEdgeL[i]);
        perm[i] = ParityLookup(c0, c1);
    }

    // South Pole edges
    for (int i = 0; i < CMegaminx::kNumSouthPoleEdges; i++)
    {
        CMegaminx::color_t c0 =
            fMega.EdgeFaceletColor(fMega.kSouthPoleEdgeN[i],
                                   fMega.kSouthPoleEdgeS[i]);
        CMegaminx::color_t c1 =
            fMega.EdgeFaceletColor(fMega.kSouthPoleEdgeS[i],
                                   fMega.kSouthPoleEdgeN[i]);
        perm[i + 5] = ParityLookup(c0, c1);
    }

    // Now figure out the parity of perm
    bool bVisitedPerm[10];
    // indexed same as perm; whether we
    // have counted that transition
    int cycleLengths = 0; // sum of (cycle length - 1)

    for (int i = 0; i < 10; i++)
        bVisitedPerm[i] = false;
    for (int i = 0; i < 10; i++)
    {
        if (bVisitedPerm[i])
            continue;

        // follow the cycle starting at perm[i]
        int next = i;
        while (!bVisitedPerm[next])
        {
            cycleLengths++;
            bVisitedPerm[next] = true;
            next = perm[next];
        }
        cycleLengths--;
    }

    bool bEvenParity = ((cycleLengths & 1) == 0);
    return bEvenParity;
}

```

```

// look up the correct Southern edge for these colors; returns
// index into SouthEq table, or index + 5 into SouthPole table
int CSolver::ParityLookup(CMegaminx::color_t c0,
CMegaminx::color_t c1)
{
    for (int i = 0; i < CMegaminx::kNumSouthEqEdges; i++)
    {
        CMegaminx::color_t trialColor0 =
            CMegaminx::CorrectColor(CMegaminx::kSouthEqEdgeL[i]);
        CMegaminx::color_t trialColor1 =
            CMegaminx::CorrectColor(CMegaminx::kSouthEqEdgeR[i]);
        if ((c0 == trialColor0 && c1 == trialColor1) ||
            (c1 == trialColor0 && c0 == trialColor1))
            return i;
    }

    for (int i = 0; i < CMegaminx::kNumSouthPoleEdges; i++)
    {
        CMegaminx::color_t trialColor0 =
            CMegaminx::CorrectColor(CMegaminx::kSouthPoleEdgeN[i]);
        CMegaminx::color_t trialColor1 =
            CMegaminx::CorrectColor(CMegaminx::kSouthPoleEdgeS[i]);
        if ((c0 == trialColor0 && c1 == trialColor1) ||
            (c1 == trialColor0 && c0 == trialColor1))
            return i + 5;
    }

    assert(false); // trouble, no match
    return 0;
}

#pragma mark == Solution Steps ==

// Solution Steps
// Solution Steps

void CSolver::Step3()
{
    Step3Edges();
    Step3Corners();

    Step3Verify();
}

void CSolver::Step3Edges()
{
    for (int i = 0; i < CMegaminx::kNumNorthPoleEdges; i++)
    {
        CMegaminx::face_t destFaceN =
            CMegaminx::kNorthPoleEdgeN[i];
        CMegaminx::face_t destFaceS =
            CMegaminx::kNorthPoleEdgeS[i];
        if (fMega.IsEdgeCorrect(destFaceN, destFaceS))
            continue; // already done!

        // if not the correct colors, find an edge that does have
        // the correct colors and drop it to the South Pole.
        // the return value is the South Equatorial face where
        // it got dropped.
        CMegaminx::color_t c0 = fMega.CorrectColor(destFaceN);
        CMegaminx::color_t c1 = fMega.CorrectColor(destFaceS);
        CMegaminx::face_t southPoleFace = Step3_4Drop(c0, c1);

        // now loft it back to the North Pole; first rotate
        // the South Pole so the edge touches the "down right" face.
        CMegaminx::face_t rotToFace =
            CMegaminx::kFaceDownRight[destFaceS];
        int dist = Distance(southPoleFace, rotToFace);

        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, dist);
        fMega.Slice(rotToFace, CMegaminx::eCW, 2);
        fMega.Slice(destFaceS, CMegaminx::eCounterCW, 2);

        // if the edge is not correctly oriented we need
        // to reorient it
        if (fMega.EdgeFaceletColor(destFaceN, destFaceS) != 1)
        {
            fMega.Slice(destFaceS, CMegaminx::eCounterCW, 2);
            int nextSouthFace = fMega.NextSouthEqFace(rotToFace);
            fMega.Slice(nextSouthFace, CMegaminx::eCW, 1);
            fMega.Slice(rotToFace, CMegaminx::eCW, 1);
            fMega.Slice(destFaceS, CMegaminx::eCounterCW, 2);
        }
    }
}

```

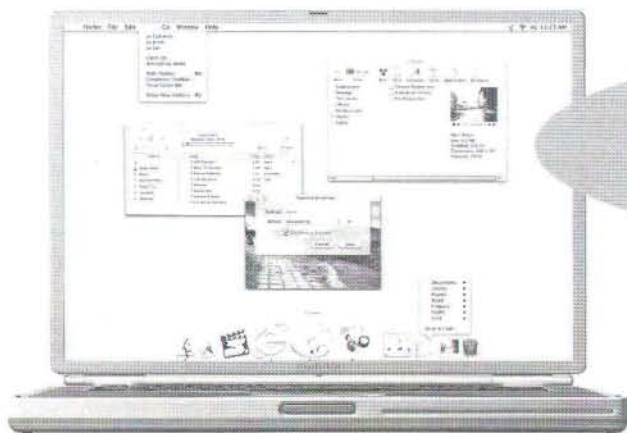
Special Small Dogs

Here are just a few of the photos our customers have sent us of their special dog friends!



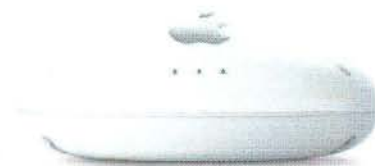
Check out all the other special Small Dog photos and send us one of your own at www.smalldog.com!

Small Dog's Special



PowerBook G4/667 Titanium
\$3199

- 512MB RAM
- 30gb Hard Drive
- Combo Drive
- Airport card and base station



Purchase from an Apple Specialist!

Small Dog Electronics is proud to have earned Apple's prestigious designation of Apple Specialist. Small Dog Electronics has exceeded all of Apple's stringent requirements for its Specialists and has taken it one step further.

Small Dog sells only the most powerful personal computers in the world—every single one of them an Apple Macintosh! In addition to the rigorous Apple training program, each Small Dog employee has chosen a specific area of expertise to concentrate upon. You can be assured that whomever you talk to at Small Dog, from our sales engineers to our shipping department, you are talking to a trained, enthusiastic, Apple Macintosh Specialist!

Small Dog Electronics is also an Authorized Apple Service Provider Plus. We have certified Apple Technicians that utilize genuine Apple parts for all repairs and provide technical support for our customers based upon their experience in upgrading and supporting thousands of Apple Macintosh computers for our customers!

Talk To an Apple Professional!

Did you know that when you call Small Dog Electronics, you are most likely talking to an Apple Product Professional? Each year, we participate in Apple's on-line courses and seminar training programs to learn as much as we can about the products that we sell and service!

Exclusive Top Dog Membership Treats

For each dollar purchased on-line at the Small Dog web site, you will receive a doggie treat. You can redeem your accumulated treats for Small Dog or Apple apparel and other products. You are automatically enrolled and begin accumulating treats in the Top Dog Club with your first purchase. Remember the more you buy, the more treats for you!



**Small Dog
Electronics**

1673 Main Street
Waitsfield, VT 05673 USA
Phone: 802-496-7171
Online: smalldog.com



```

    }
}

// returns face we dropped it to
CMegaminx::face_t CSolver::Step3_4Drop(CMegaminx::color_t c0,
    CMegaminx::color_t c1)
{
    fMega.WriteComment("Step3_4Drop");
    // search the lower edges for one having these colors
    // (in either order), and if found move it to
    // the South Pole.

    bool bFound = false;
    CMegaminx::face_t southPoleFace = 0;
    // edge dropped to this face-SouthPole

    // search the South Pole edges, and if found we are done!
    if (!bFound)
    {
        for (int i = 0; i < CMegaminx::kNumSouthPoleEdges &&
            !bFound; i++)
        {
            CMegaminx::face_t trialFaceN =
                CMegaminx::kSouthPoleEdgeN[i];
            CMegaminx::face_t trialFaceS =
                CMegaminx::kSouthPoleEdgeS[i];
            if (fMega.EdgeHasColors(trialFaceN, trialFaceS, c0,
                c1))
            {
                fMega.WriteComment("Step3_4Drop found on South
                Pole");
                bFound = true;
                southPoleFace = trialFaceN;
            }
        }

        // search the South Equator edges, and if found drop
        // the edge to the South Pole by rotating its left face
        // CW 1
        if (!bFound)
        {
            for (int i = 0; i < CMegaminx::kNumSouthEqEdges &&
                !bFound; i++)
            {
                CMegaminx::face_t trialFaceL =
                    CMegaminx::kSouthEqEdgeL[i];
                CMegaminx::face_t trialFaceR =
                    CMegaminx::kSouthEqEdgeR[i];
                if (fMega.EdgeHasColors(trialFaceL, trialFaceR, c0,
                    c1))
                {
                    fMega.WriteComment("Step3_4Drop found on South
                    Equator");
                    bFound = true;
                    fMega.Slice(trialFaceL, CMegaminx::eCW, 1);
                    southPoleFace = trialFaceL;
                }
            }

            // search the Middle Equator edges, and if found drop
            // the edge to the South Pole by rotating its S face
            // either CW 2 or CCW 2
            if (!bFound)
            {
                for (int i = 0; i < CMegaminx::kNumMiddleEqEdges &&
                    !bFound; i++)
                {
                    CMegaminx::face_t trialFaceN =
                        CMegaminx::kMiddleEqEdgeN[i];
                    CMegaminx::face_t trialFaceS =
                        CMegaminx::kMiddleEqEdgeS[i];
                    if (fMega.EdgeHasColors(trialFaceN, trialFaceS, c0,
                        c1))
                    {
                        fMega.WriteComment("Step3_4Drop found on Middle
                        Equator");
                    }
                }
            }
        }
    }
}

```

```

    bFound = true;
    southPoleFace = trialFaceS;
    // even indices are below and right of N face,
    // therefore above and left of S face
    if ((i & 1) == 0)
    {
        // above and left, so use CCW 2
        fMega.Slice(trialFaceS, CMegaminx::eCounterCW, 2);
    }
    else
    {
        // above and right, so use CW 2
        fMega.Slice(trialFaceS, CMegaminx::eCW, 2);
    }
}

// search the North Equator edges, and if found there drop to the
// South Pole. The Meffert solution uses a simple transformation
// in case 3 and a complicated one in case 4 (where it has to avoid
// disturbing other North Equator edges), but we will use the
// complicated one in both cases because the implementation
// is easier.
if (!bFound)
{
    for (int i = 0; i < CMegaminx::kNumNorthEqEdges; i++)
    {
        CMegaminx::face_t trialFaceL =
            CMegaminx::kNorthEqEdgeL[i];
        CMegaminx::face_t trialFaceR =
            CMegaminx::kNorthEqEdgeR[i];
        if (fMega.EdgeHasColors(trialFaceL, trialFaceR, c0,
            c1))
        {
            fMega.WriteComment("Step3_4Drop found on North
            Equator");
            bFound = true;

            // drop to down left
            southPoleFace =
                CMegaminx::kFaceDownLeft[trialFaceL];

            fMega.Slice(southPoleFace, CMegaminx::eCounterCW,
                1);
            DoLUU(trialFaceL, trialFaceR);
            DoLUU(trialFaceL, trialFaceR);
            fMega.Slice(southPoleFace, CMegaminx::eCW, 1);
            DoRUU(trialFaceL, trialFaceR);
            DoRUU(trialFaceL, trialFaceR);

            // at this point the edge is at the upper left of
            // southPoleFace, so rotate it to put it on the
            // South Pole
            fMega.Slice(southPoleFace, CMegaminx::eCounterCW,
                2);
        }
    }

    // search the North Pole edges, and if found drop to the
    // South Pole. (This code should only be execute for Step 3,
    // because in Step 4 the North Pole edges have already been
    // set and we should have found the desired edge before now.)
    if (!bFound)
    {
        for (int i = 0; i < CMegaminx::kNumNorthPoleEdges; i++)
        {
            CMegaminx::face_t trialFaceN =
                CMegaminx::kNorthPoleEdgeN[i];
            CMegaminx::face_t trialFaceS =
                CMegaminx::kNorthPoleEdgeS[i];
            if (fMega.EdgeHasColors(trialFaceN, trialFaceS, c0,
                c1))
            {
                fMega.WriteComment("Step3_4Drop found on North
                Pole");
                bFound = true;

                southPoleFace =
                    CMegaminx::kFaceDownRight[trialFaceS];
                fMega.Slice(trialFaceS, CMegaminx::eCW, 2);
            }
        }
    }
}

```

A photograph of a wooden door with a silver handle. A yellow sticky note is pinned to the door. The note has handwritten text in black ink.

Dear Burglar,

The key is under
the mat. Beer and
sandwiches are
in the fridge.

♥ Greg & Patty

We're Easier.

Create anything from prototypes to full professional applications. Just drag and drop interface elements while REALbasic handles the details. You concentrate on what makes your stuff great — your ideas! REALbasic compiles native applications for Macintosh, Mac OS X and Windows without platform-specific adjustments. It's the powerful, easy-to-use tool for creating your own software. Each version of your software looks and works just as it should in each environment.

Complex problems shouldn't require complex solutions. The answer is REALbasic.



Download a free demo. www.realbasic.com

```

    fMega.Slice(southPoleFace, CMegaminx::eCounterCW,
2);
    }
}

assert(bFound);

return southPoleFace;
}

void CSolver::Step3Corners()
{
    for (CMegaminx::vertex_t destCorner = 0; destCorner < 5;
        destCorner++)
    {
        // maybe corner is already done!
        if (fMega.IsCornerCorrect(destCorner))
            continue;

        fMega.WriteComment("Step3Corners");
        // find the corner that should be here, and drop
        // it to the South Pole and move it into place.
        CMegaminx::color_t destc0 =

fMega.CorrectColor(CMegaminx::kCornerFaces[destCorner][0]);
        CMegaminx::color_t destc1 =

fMega.CorrectColor(CMegaminx::kCornerFaces[destCorner][1]);
        CMegaminx::color_t destc2 =

fMega.CorrectColor(CMegaminx::kCornerFaces[destCorner][2]);
        CMegaminx::vertex_t srcCorner =
        fMega.CornerHavingColors(destc0, destc1, destc2);

        // special transformation if the src is at the
        // North Pole
        if (fMega.IsNorthPoleVertex(srcCorner))
        {
            fMega.WriteComment("Step3Corners drop North Pole
corner");
            CMegaminx::face_t faceL =
                CMegaminx::kCornerFaces[srcCorner][1];
            CMegaminx::face_t faceR =
                CMegaminx::kCornerFaces[srcCorner][2];
            DoRUU(faceL, faceR);
            srcCorner += 5; // corner has dropped to North Equator
        }

        // drop the corner to the South Pole (if it is not
        // already there)
        if (fMega.IsNorthEquatorVertex(srcCorner))
        {
            fMega.Slice(CMegaminx::kFaceBelow[srcCorner],
                CMegaminx::eCW, 2);
            srcCorner += 10; // corner has dropped to South Pole
        }
        else if (fMega.IsSouthEquatorVertex(srcCorner))
        {
            fMega.Slice(CMegaminx::kFaceBelow[srcCorner],
                CMegaminx::eCW, 1);
            srcCorner += 5;
        }

        // rotate the vertex into place
        int moveToCorner = destCorner + 15;
        int dist = Distance(srcCorner, moveToCorner);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCW, dist);

        // lift the vertex into place on the North Equator
        int bottomFace = CMegaminx::kFaceBelow[destCorner + 5];
        fMega.Slice(bottomFace, CMegaminx::eCounterCW, 2);

        // figure out the orientation and apply the correct
        // transformation to lift it to the North Pole
        CMegaminx::face_t leftFace =
            CMegaminx::kFaceBelow[destCorner];
        CMegaminx::face_t rightFace =
            CMegaminx::PrevNorthEqFace(leftFace);
        if (fMega.CornerFaceletColor(leftFace, rightFace,
            bottomFace)
            == 1)
    {

```

```

        // top color at left
        // NOTE: Meffert solution wrongly states to use
        // LUU in this case.
        DoRUU(leftFace, rightFace);
    }
    else if (fMega.CornerFaceletColor(rightFace, leftFace,
        bottomFace) == 1)
    {
        // top color at right
        DoLUU(leftFace, rightFace);
    }
    else
    {
        // top color at bottom
        DoLUU(leftFace, rightFace);
        DoLUU(leftFace, rightFace);
        DoLUU(leftFace, rightFace);
    }
}

////////////////////////////////////
// Step 4. Setting the northern equatorial edges
////////////////////////////////////
//
// Very similar to Step 3 edge case; the common 3_4 routine does
// most of the work.

void CSolver::Step4()
{
    for (int i = 0; i < CMegaminx::kNumNorthEqEdges; i++)
    {
        CMegaminx::face_t destFaceL =
            CMegaminx::kNorthEqEdgeL[i];
        CMegaminx::face_t destFaceR =
            CMegaminx::kNorthEqEdgeR[i];
        if (fMega.IsEdgeCorrect(destFaceL, destFaceR))
            continue; // already done!

        // if not the correct colors, find an edge that does have
        // the correct colors and drop it to the South Pole.
        // the return value is the South Equatorial face where
        // it got dropped.
        CMegaminx::color_t c0 = fMega.CorrectColor(destFaceL);
        CMegaminx::color_t c1 = fMega.CorrectColor(destFaceR);
        CMegaminx::face_t southPoleFace = Step3_4Drop(c0, c1);

        // now lift it back to the North Equator
        // figure the face we want to be under, and
        // rotate the South Pole to get there. The
        // desired face lies directly underneath the
        // desired edge, and therefore below and left of
        // the destFaceR.
        CMegaminx::face_t rotToFace =
            CMegaminx::kFaceDownLeft[destFaceR];
        int dist = Distance(southPoleFace, rotToFace);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, dist);

        // rotate rotToFace either CW 2 or CCW 2 to bring
        // the edge adjacent faceL or faceR; we pick the
        // rotation so that the facelet on the face
        // has the face color. This facelet is currently
        // on the South Pole face.
        // Finally we'll move it into the correct edge.
        //
        // We combine the CW 2 and CCW 1 to get CW 1, and
        // similarly.
        int faceletColor = fMega.EdgeFaceletColor(kSouthPoleFace,
            rotToFace);
        if (faceletColor == destFaceL)
        {
            fMega.Slice(rotToFace, CMegaminx::eCW, 1); // = CW 2
            and CCW 1
            DoLUU(destFaceL, destFaceR);
            DoLUU(destFaceL, destFaceR);
            fMega.Slice(rotToFace, CMegaminx::eCW, 1);
            DoRUU(destFaceL, destFaceR);
            DoRUU(destFaceL, destFaceR);
        }
        else
        {
            fMega.Slice(rotToFace, CMegaminx::eCounterCW, 1);
            // = CCW 2 and CW 1
            DoRUU(destFaceL, destFaceR);

```

Tri-platform Calendaring

OS X *Carbon*

Mac OS *7.1-9.x*

Windows *95-2000*

New!

```
DoRUU(destFaceL, destFaceR);
fMega.Slice(rotToFace, CMegaminx::eCounterCW, 1);
DoLUU(destFaceL, destFaceR);
DoLUU(destFaceL, destFaceR);
}

Step4Verify():
}

////////////////////////////////////////
// Step 5. Setting the northern equatorial corners
////////////////////////////////////////
//
// We step through the vertices, finding the correctly-oriented
// corner that belongs there. To transfer the corner, drop it to
// the South Pole, rotate, then rotate up to the North Equator.
void CSolver::Step5()
{
    for (int destVertex = CMegaminx::kFirstNorthEqVertex;
         destVertex <= CMegaminx::kLastNorthEqVertex;
         destVertex++)
    {
        if (fMega.IsCornerCorrect(destVertex))
            continue; // already OK, skip this one

        // Find the corner whose colors should be moved here. This
        // may be the same corner, if it is not oriented correctly.
        int c0 =

fMega.CorrectColor(CMegaminx::kCornerFaces[destVertex][0]);
int c1 =

fMega.CorrectColor(CMegaminx::kCornerFaces[destVertex][1]);
int c2 =

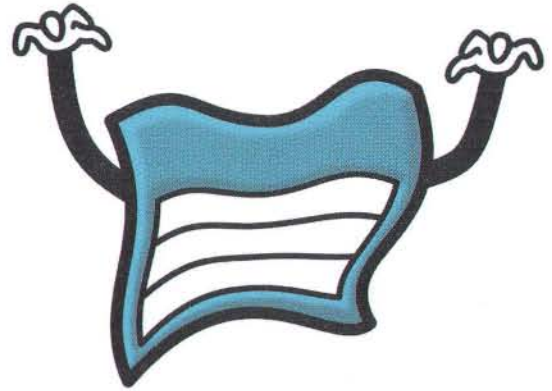
fMega.CorrectColor(CMegaminx::kCornerFaces[destVertex][2]);
int srcVertex = fMega.CornerHavingColors(c0, c1, c2);
if (srcVertex != destVertex)
    Step5PlaceVertex(srcVertex, destVertex);
    Step5OrientVertex(destVertex);
    }
    Step5Verify();
}

// place and position the srcVertex into the destVertex
void CSolver::Step5PlaceVertex(int srcVertex, int destVertex)
{
    // drop the src to the South Pole if needed
    int southPoleFromVertex = srcVertex;
    if (fMega.IsNorthEquatorVertex(srcVertex))
    {
        fMega.WriteComment("Step5PlaceVertex from North
Equator");
        southPoleFromVertex = srcVertex + 10;
        fMega.Slice(CMegaminx::kFaceBelow[srcVertex],
                    CMegaminx::eCW, 2);
    }
    else if (fMega.IsSouthEquatorVertex(srcVertex))
    {
        // moving this vertex also disturbs the North Equator
        // vertex on this face, which might already be correctly
        // placed, so we must rotate the face back after all
        // movements are done. We will handle this by
        // rotating the South Pole face CounterCW by 1 and
        // then reversing the face rotation.
        fMega.WriteComment("Step5PlaceVertex from South
Equator");
        southPoleFromVertex = srcVertex + 5;
        int rot2Face = CMegaminx::kFaceBelow[srcVertex];
        fMega.Slice(rot2Face, CMegaminx::eCW, 1);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
        fMega.Slice(rot2Face, CMegaminx::eCounterCW, 1);
    }

    // figure where we need to rotate South Pole to, and the
    // face to rotate CounterCW to left to final position
    fMega.WriteComment("Step5PlaceVertex move vertex into
place");
    int southPoleToVertex = destVertex + 10;
```

CalendarMonster 1.2

Multi-platform calendar



- **End-user Calendar**
The best FMP calendar experience
 - Fast, fast, fast!!!
 - Platform-appropriate user experience
 - Highly scalable
- **Excellent Developer Support**
Quickly and easily integrate with your FileMaker Pro projects
 - 100% live data, 100% modular, no plug-ins
 - FileMaker Pro 4 or 5 (including 5.5)
 - As little as 2 ScriptMaker calls
 - Dedicated developers' utility



Developer and volume licensing available



**ASCENDING
TECHNOLOGIES**

monster@asctech.com

Download your own FREE fully-functional demo
<http://www.asctech.com>

```

// rotate the South Pole CW into position
int dist = Distance(southPoleFromVertex,
southPoleToVertex);
fMega.Slice(kSouthPoleFace, CMegaminx::eCW, dist);

// raise the src into the dest
int homeFace = CMegaminx::kFaceBelow[destVertex];
fMega.Slice(homeFace, CMegaminx::eCounterCW, 2);
}

void CSolver::Step5OrientVertex(int destVertex)
{
    fMega.WriteComment("Step5OrientVertex");
    // orient the corner, if needed
    // figure the colors of the corner facelets and see if
    // we need to rotate the corner
    CMegaminx::face_t belowFace =
        CMegaminx::kFaceBelow[destVertex];

    CMegaminx::color_t belowColor =
fMega.CorrectColor(belowFace);
    // color of bottom face
    CMegaminx::face_t rightFace =
        CMegaminx::kFaceAbove[destVertex];
    CMegaminx::face_t leftFace =
        CMegaminx::NextNorthEqFace(rightFace);
    if (fMega.CornerFaceletColor(leftFace, rightFace,
belowFace) ==
        belowColor)
    {
        fMega.Slice(belowFace, CMegaminx::eCW, 2);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
        fMega.Slice(belowFace, CMegaminx::eCW, 2);
    }
    else if (fMega.CornerFaceletColor(rightFace, belowFace,
leftFace) == belowColor)
    {
        fMega.Slice(belowFace, CMegaminx::eCounterCW, 2);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);
        fMega.Slice(belowFace, CMegaminx::eCounterCW, 2);
    }
}

////////////////////////////////////
// Step 6. Setting the middle equatorial edges
////////////////////////////////////
//
// We step through the middle equatorial edges, checking to see if
// each already has the correctly positioned and placed edge, and if
// not then searching the South Pole edges, the South Equatorial
// edges, and finally the middle equatorial edges (after this one)
// for the needed edge. Note that each combination of colors has
// two edges with this combination, and (I think) they are
// interchangeable; this is unlike the situation for corners, where
// there are also two corners with a given combination, but they
// have opposite orientations and are not interchangeable.
void CSolver::Step6()
{
    for (int destFaceIndex = 0;
        destFaceIndex < CMegaminx::kNumMiddleEqEdges;
        destFaceIndex++)
    {
        CMegaminx::face_t faceS =
            CMegaminx::kMiddleEqEdgeS[destFaceIndex];
        CMegaminx::face_t faceN =
            CMegaminx::kMiddleEqEdgeN[destFaceIndex];
        if (fMega.IsEdgeCorrect(faceS, faceN))
            continue; // already correctly placed and positioned
        CMegaminx::color_t neededColorS =
fMega.CorrectColor(faceS);
        CMegaminx::color_t neededColorN =
fMega.CorrectColor(faceN);
        bool bFound = false;

        // search the polar edges
        for (int i = 0;
            i < CMegaminx::kNumSouthPoleEdges && !bFound; i++)
        {
            CMegaminx::face_t searchPoleFace =
                CMegaminx::kSouthPoleEdgeN[i];
            if (fMega.EdgeHasColors(kSouthPoleFace, searchPoleFace,
neededColorS, neededColorN))

```

```

{
    bFound = true;
    fMega.WriteComment("Step6 move from South Pole");
    Step6PlacePoleEdge(searchPoleFace, destFaceIndex);
}

// search the Southern Equatorial edges
for (int i = 0;
    i < CMegaminx::kNumSouthEqEdges && !bFound; i++)
{
    CMegaminx::face_t searchEqFaceL =
        CMegaminx::kSouthEqEdgeL[i];
    CMegaminx::face_t searchEqFaceR =
        CMegaminx::kSouthEqEdgeR[i];
    if (fMega.EdgeHasColors(searchEqFaceL, searchEqFaceR,
neededColorS, neededColorN))
    {
        bFound = true;
        fMega.WriteComment("Step6 move from South
Equatorial");
        DoRLL(searchEqFaceR, searchEqFaceL);
        Step6PlacePoleEdge(searchEqFaceR, destFaceIndex);
    }

    // search the (this or later) middle equatorial edges
    // we don't search earlier ones because they are already
    // correctly placed and we don't want to steal from them;
    // we do need to search the edge itself because it might
    // have the correct colors but wrongly placed.
    for (int searchIndex = destFaceIndex;
        searchIndex < CMegaminx::kNumMiddleEqEdges &&
!bFound;
        searchIndex++)
    {
        CMegaminx::face_t mFaceS =
            CMegaminx::kMiddleEqEdgeS[searchIndex];
        CMegaminx::face_t mFaceN =
            CMegaminx::kMiddleEqEdgeN[searchIndex];
        if (fMega.EdgeHasColors(mFaceS, mFaceN, neededColorS,
neededColorN))
        {
            bFound = true;
            fMega.WriteComment("Step6 move from Middle
Equatorial");
            // lift the found edge, either right or left.
            // Lifting uses the same transformations as
            // dropping, however the lifted edge goes to the
            // adjoining face.
            if ((searchIndex & 1) == 0)
            {
                // even index, so edge is below and to right,
                // and will be lifted to next face
                CMegaminx::face_t nextMFace =
                    fMega.NextSouthEqFace(mFaceS);
                DoLUU(mFaceS, nextMFace);
                DoLLL(mFaceS, nextMFace);
                DoRUU(mFaceS, nextMFace);
                Step6PlacePoleEdge(nextMFace, destFaceIndex);
            }
            else
            {
                // odd index, so edge is below and to left,
                // and will be lifted to previous face
                CMegaminx::face_t prevFace =
                    fMega.PrevSouthEqFace(mFaceS);
                DoRUU(prevFace, mFaceS);
                DoRLL(prevFace, mFaceS);
                DoLUU(prevFace, mFaceS);
                Step6PlacePoleEdge(prevFace, destFaceIndex);
            }
        }
    }
    assert(bFound);
}

bool bEdgeParityOK = CheckEdgeParity();
if (!bEdgeParityOK)
{
    // take evasive action; we will swap two same-colored
    // edges in the equator. This is a transposition, so
    // it should cause the edges in the South half to

```

Classic CARBON
COCOA Mac OS 9

Mac OS X

Where Do You Get
Your Source?

MacTech

MacTech

The Database
Learn to leverage
list a



MacTech®

Mac OS X:
CARBON
COCOA



MacTech®

Mac OS X
Beta Review



Do What
The EXPERTS Do....

Read MacTech®

Subscribe
TODAY!

www.mactech.com

PO Box 5200 • Westlake Village, CA • 91359-5200
800/MACDEV-1 (800/622-3381) • Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

```

// switch to even parity. We'll somewhat arbitrarily
// swap the two 2-4 color edges, located at 2-10
// and 4-8. Just as in earlier Step 6 work we move one
// edge to the South Pole, place it correctly which
// moves the other edge to the South Pole, then place
// that edge.

fMega.WriteComment("Step6 evasive action to fix
parity");
DoLUU(10, 11); // move 2-10 to South Pole
DoLLL(10, 11);
DoRUU(10, 11);
fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 2);
// position
DoRUU(12, 8); // move 2-10 to equator, 4-8 to South Pole
DoRLL(12, 8);
DoLUU(12, 8);
fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 2); // position
DoLUU(10, 11); // move 4-8 to equator
DoLLL(10, 11);
DoRUU(10, 11);
}

```

```
Step6Verify();
```

```

void CSolver::Step6PlacePoleEdge(int fromSFace, int
toEdgeIndex)
{
// rotate the edge CounterCW to the correct position
fMega.WriteComment("Step6PlacePoleEdge");
CMegaminx::face_t toSFace =
CMegaminx::kMiddleEqEdgeS[toEdgeIndex];
int dist = Distance(fromSFace, toSFace);
fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, dist);

```

```

// flip the edge if it is wrongly oriented
CMegaminx::face_t nextFace =
fMega.NextSouthEqFace(toSFace);
if (fMega.EdgeFaceletColor(toSFace, kSouthPoleFace) !=
fMega.CorrectColor(toSFace))
{
fMega.WriteComment("Step6PlacePoleEdge flip edge");
DoRLL(toSFace, nextFace);
fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);
}

```

```
// now drop it into position, either on left or right
```

```

CMegaminx::face_t prevFace =
fMega.PrevSouthEqFace(toSFace);
if ((toEdgeIndex & 1) == 0)
{
// even index, so edge is below and to right
DoLUU(toSFace, nextFace);
DoLLL(toSFace, nextFace);
DoRUU(toSFace, nextFace);
}
else
{
// odd index, so edge is below and to left
DoRUU(prevFace, toSFace);
DoRLL(prevFace, toSFace);
DoLUU(prevFace, toSFace);
}
}

```

```

// Step 7. Setting the Southern Equatorial Edges
//

```

```

void CSolver::Step7()
{
for (int i = 0; i < CMegaminx::kNumSouthEqEdges; i++)
{
CMegaminx::face_t destFaceL =
CMegaminx::kSouthEqEdgeL[i];
CMegaminx::face_t destFaceR =
CMegaminx::kSouthEqEdgeR[i];
if (fMega.IsEdgeCorrect(destFaceL, destFaceR))
continue;

CMegaminx::color_t destColorL =
fMega.CorrectColor(destFaceL);
CMegaminx::color_t destColorR =

```

```
fMega.CorrectColor(destFaceR);
```

```

// check to see if needed color is on South Pole
bool bFound = false;
for (int j = 0; j < CMegaminx::kNumSouthPoleEdges &&
!bFound;
j++)
{
CMegaminx::face_t srcFaceN =
CMegaminx::kSouthPoleEdgeN[j];
CMegaminx::face_t srcFaceS =
CMegaminx::kSouthPoleEdgeS[j];
if (fMega.EdgeHasColors(srcFaceN, srcFaceS,
destColorL,
destColorR))
{
bFound = true;
Step7PlacePoleEdge(srcFaceN, destFaceL, destFaceR);
}
}

```

```

// check if needed color is on South Equator; do not
// check already-placed edges
if (!bFound)
{
for (int j = 1; j < CMegaminx::kNumSouthEqEdges &&
!bFound;
j++)
{

```

```

int srcFaceL = CMegaminx::kSouthEqEdgeL[j];
int srcFaceR = CMegaminx::kSouthEqEdgeR[j];
if (fMega.EdgeHasColors(srcFaceL, srcFaceR,
destColorL,
destColorR))
{
// loft the edge using RLL, so it goes above srcFaceR,
// then move to correct place (remember that we are
// looking at the Megaminx upside down, so the
// left face is srcFaceR)
bFound = true;
fMega.WriteComment("Step7 loft edge");
DoRLL(srcFaceR, srcFaceL);
Step7PlacePoleEdge(srcFaceR, destFaceL,
destFaceR);
}
}
assert(bFound);
}
Step7Verify();
}

```

```

// place an equatorial edge that is currently on the pole;
// eqFace is the equatorial face it is below.

```

```

void CSolver::Step7PlacePoleEdge(CMegaminx::face_t srcFaceN,
CMegaminx::face_t destFaceL,
CMegaminx::face_t destFaceR)
{

```

```

// find the face it belongs to and rotate it there;
// find the CW distance we should move; we move it so
// its equatorial color matches the destination face color. Then
// lift it into position using RLL or LLL. Remember we measure
// right and left with the Megaminx right-side up.
fMega.WriteComment("Step7PlacePoleEdge");
CMegaminx::color_t destFaceColor =
fMega.EdgeFaceletColor(srcFaceN, kSouthPoleFace);
bool bLiftFromLeft =
(destFaceColor == fMega.CorrectColor(destFaceL));
CMegaminx::face_t destFace =
bLiftFromLeft ? destFaceL : destFaceR;
int dist = Distance(destFace, srcFaceN);
fMega.Slice(kSouthPoleFace, CMegaminx::eCW, dist);
if (bLiftFromLeft)
DoRLL(destFaceR, destFaceL);
else
DoLLL(destFaceR, destFaceL);
}

```

```

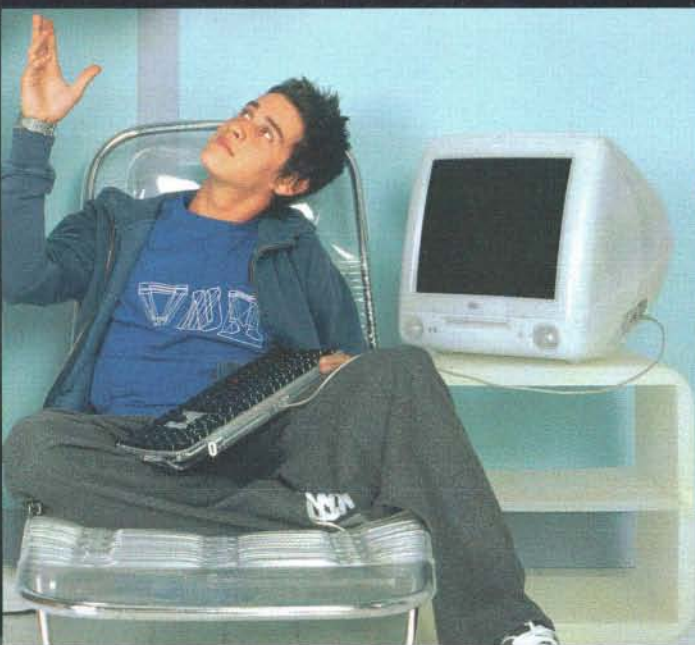
// Step 8. Setting the South Pole edges
//

```

```

// This step both positions and orients the South Pole edges.
//

```



MacDirectory™

MACDIRECTORY MAGAZINE

CREATIVE DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS AND OUR EXPERT TECHNOLOGY TEAM OFFER THEIR OWN PERSONAL INTERPRETATION OF THINGS THAT ONLY THE MACINTOSH SYSTEM CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, PRODUCT NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH BUYERS' GUIDE - INCLUDING OVER 5,000 PRODUCTS AND SERVICES FOR YOUR MAC.

INTERVIEWS

TAPPING INTO THE WORLD OF CELEBRITIES AND THEIR MACS, ONLY MACDIRECTORY OFFERS IN-DEPTH INTERVIEWS. GET A CLOSE AND PERSONAL VIEW ON STEVE JOBS, TOM CRUISE, CLAUDIA SCHIFFER, MADONNA, GEORGE LUCAS AND OTHER LEADERS IN THE MAC COMMUNITY.

CULTURE

MACDIRECTORY TAKES YOU TO THE WILDEST CORNERS OF THE GLOBE AND UNCOVERS MANY USES OF MACINTOSH COMPUTERS. TRAVEL TO JAPAN, AUSTRALIA, GERMANY, INDIA, BRAZIL, RUSSIA AND LEARN MORE ABOUT APPLE'S CULTURAL IMPACT AROUND THE GLOBE.

SUBSCRIBE

SUBSCRIBE ONLINE FOR FASTER DELIVERY: www.macdirectory.com/sub.html
SUBSCRIPTION RATES FOLLOWS: 1 YEAR, \$32.00 FOR 4 ISSUES & 2 YEARS, \$60.00 FOR 8 ISSUES. OR YOU MAY ALSO MAIL A CHECK OR MONEY ORDER TO: MACDIRECTORY
SUBSCRIPTION DEPT. 150 WEST 25TH ST NY NY 10001. INCLUDE YOUR PHONE, EMAIL AND MAILING ADDRESS.

```

// We pick a fixed orientation to make the rotation calculations easy.
// The parked edge is on faces 8 and 9, and we rotate it for lofting
// to be on faces 7 and 8, so we'll use LLL to loft it.
// The reference edge is on faces 7 and 10, the second edge is on faces
// 7 and 11, and the third edge is on faces 7 and 12.
//
// NOTE: All edge operations must be done using the fixed
// edge 8-9, otherwise things won't be properly aligned
// after setting the first 3 edges.
//
// We don't have to return the South Pole after each move.

void CSolver::Step8()
{
    Step8ReferenceEdge();
    Step8SecondEdge();
    Step8ThirdEdge();
    Step8RestoreEquator();
    Step8OrientFourFive();

    Step8Verify();
}

void CSolver::Step8ReferenceEdge()
{
    if (fMega.IsEdgeCorrect(7, 10))
        return; // already correct, no action needed

    fMega.WriteComment("Step8ReferenceEdge");
    // place and orient the reference edge

    // locate the correctly colored edge
    bool bFound = false;
    CMegaminx::face_t srcFace = 0;
    for (int i = 0; i < CMegaminx::kNumSouthPoleEdges &&
!bFound;
        i++)
    {
        srcFace = CMegaminx::kSouthPoleEdgeN[i];
        bFound = fMega.EdgeHasColors(srcFace, kSouthPoleFace, 1,
4);
    }
    assert(bFound);

    if (fMega.EdgeFaceletColor(kSouthPoleFace, srcFace) != 1)
    {
        // need to orient edge
        // first move the edge over to flipping area, at face 9
        int dist = Distance(9, srcFace);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCW, dist);

        // now flip the edge; it will go to face 8
        DoLLL(8, 9);
        srcFace = 8; // pretend it was here all along
    }

    // move the edge into position at edge 10
    int dist = Distance(10, srcFace);
    fMega.Slice(kSouthPoleFace, CMegaminx::eCW, dist);
}

void CSolver::Step8SecondEdge()
{
    if (fMega.IsEdgeCorrect(7, 11))
        return; // already correct, no action needed

    // place and orient the second edge
    // For this operation we have to return the South Pole face
    // to its original position so that the reference edge will
    // be in place.

    // locate the correct color
    bool bFound = false;
    CMegaminx::face_t srcFace = 0;
    for (int i = 0; i < CMegaminx::kNumSouthPoleEdges &&
!bFound;
        i++)
    {
        srcFace = CMegaminx::kSouthPoleEdgeN[i];
        bFound = fMega.EdgeHasColors(srcFace, kSouthPoleFace, 1,
5);
    }
    // if not found, the desired edge is already parked, so
    // skip the parking

```

```

if(bFound)
{
    // we will loft using faces 8 and 9, so the South Pole
    // face must be rotated to place aFace in one of these
    // positions, but such that the reference edge (face 10)
    // does not go to either; this means our CW rotations must
    // be something other than 1 and 2.
    bool bUseRLL = true;
    int dist = Distance(9, srcFace);
    if (dist == 2) // dist == 1 is impossible because that moves 10 to 9
    {
        dist = 3; // rotate to 10 instead
        bUseRLL = false;
    }

    fMega.WriteComment("Step8SecondEdge parking");
    {
        CTempRotate rot1(fMega, kSouthPoleFace, dist,
            CMegaminx::eCW);
        if (bUseRLL) // park edge 2
            DoRLL(8, 9);
        else
            DoLLL(8, 9);
    }

    // now move the edge from the parked position to the South Pole
    fMega.WriteComment("Step8SecondEdge placing");
    {
        CTempRotate rot2(fMega, kSouthPoleFace, 2,
            CMegaminx::eCounterCW);
        DoRLL(8, 9); // places the edge

        // if the edge is not oriented correctly, re-orient it
        if (fMega.EdgeFaceletColor(kSouthPoleFace, 8) != 1)
        {
            fMega.WriteComment("Step8SecondEdge re-orienting");
            DoRLL(8, 9);
            fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);
            DoLLL(8, 9);
            fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
            DoRLL(8, 9);
        }
    }

    void CSolver::Step8ThirdEdge()
    {
        if (fMega.IsEdgeCorrect(7, 12))
            return; // already correct, no action needed

        // place and orient the third edge
        // For this operation we have to return the South Pole face
        // to its original position so that the reference edge will
        // be in place.

        // locate the correct color
        bool bFound = false;
        CMegaminx::face_t srcFace = 0;
        for (int i = 0; i < CMegaminx::kNumSouthPoleEdges &&
!bFound;
            i++)
        {
            srcFace = CMegaminx::kSouthPoleEdgeN[i];
            bFound = fMega.EdgeHasColors(srcFace, kSouthPoleFace, 1,
6);
        }

        // if not found, the desired edge is already parked, so
        // skip the parking

        if(bFound)
        {
            // we will loft using faces 8 and 9, so the South Pole
            // face must be rotated to place aFace in one of these
            // positions, but such that the reference edge (face 10)
            // and second edge do not go there either.
            bool bUseRLL = true;
            int dist = 0;
            switch (srcFace)
            {
                case 12:
                    {

```

The Key is in your hands!

Does your dongle do all this?

Software Goes Online

Electronic Software Distribution – safely protected by WIBU-KEY.

Web Remote Programming

Reprogram the WIBU-BOX hardware at the customer's site directly via the Internet.

Web Authentication

Secure authentication via a two-way-encryption.

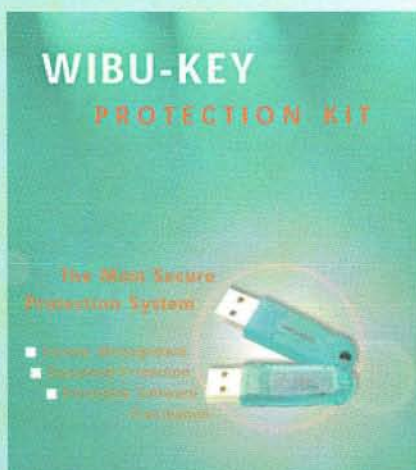
Pay-Per-Use

Usage dependent accounting.

Mac OS 9 & X

WIBU-KEY supports Mac OS, Windows and heterogeneous networks.

► **Yes? Then you are already using WIBU-KEY!**

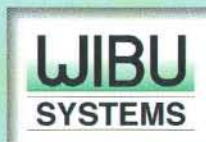


► **No? Then order your Test Kit
at 1-800-986-6578**

**You will find more information at
www.griftech.com**



USA, Canada: Griffin Technologies, LLC
phone: (785) 832-2070 · fax: (785) 832-8787
email: sales@griftech.com · www.griftech.com



WIBU-SYSTEMS AG
76137 Karlsruhe, Germany
WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
email: info@wibu.com

www.wibu.com

Test Kits also available at:

Argentina info@safeld-group.com, Australia simone.eckerle@wibu.com, Belgium wibu@impakt.be, Croatia aries@aries.hr, Denmark ivha@danbit.dk,
Finland finebyte@finebyte.com, France info@neol.fr, Hungary info@mrsoft.hu, Japan info@suncarla.co.jp, Jordan, Lebanon starsoft@cyberia.net.lb,
Korea dhkimm@wibu.co.kr, Netherlands wibu@impakt.be, Portugal dubit@dubit.pt, Syria starsoft@cyberia.net.lb, Thailand preecha@dptf-th.com,
United Kingdom info@codework.com, USA sales@griftech.com

```

// rotate CounterCW 1 to face 8, use LLL
dist = 1;
bUseRLL = false;
}
break;

case 8:
{
    // already in place on face 8, use LLL
    dist = 0;
    bUseRLL = false;
}
break;

case 9:
{
    // already in place on face 9, use RLL
    dist = 0;
    bUseRLL = true;
}
break;

default:
{
    assert(false);
}
break;
}

fMega.WriteComment("Step8ThirdEdge parking");
{
    CTempRotate rot1(fMega, kSouthPoleFace, dist,
        CMegaminx::eCounterCW);
    if (bUseRLL)
        DoRLL(8, 9);
    else
        DoLLL(8, 9);
}

// now move the edge from the parked position to the South Pole
fMega.WriteComment("Step8ThirdEdge placing");
{
    CTempRotate rot2(fMega, kSouthPoleFace, 1,
        CMegaminx::eCounterCW);
    DoRLL(8, 9); // places the edge

    // if the edge is not oriented correctly, re-orient it
    if (fMega.EdgeFaceletColor(kSouthPoleFace, 8) != 1)
    {
        DoRLL(8, 9);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);
        DoLLL(8, 9);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
        DoRLL(8, 9);
    }
}

void CSolver::Step8RestoreEquator()
{
    // figure out which South Pole edge has the equatorial edge
    // and restore it to its correct place

    fMega.WriteComment("Step8RestoreEquator");
    if (fMega.EdgeFaceletColor(kSouthPoleFace, 8) != 1 &&
        fMega.EdgeFaceletColor(8, kSouthPoleFace) != 1)
        DoLLL(8, 9); // 7-8 edge should be on equator
    else if (fMega.EdgeFaceletColor(kSouthPoleFace, 9) != 1 &&
        fMega.EdgeFaceletColor(9, kSouthPoleFace) != 1)
        DoRLL(8, 9); // 7-9 edge should be on equator
}

void CSolver::Step8OrientFourFive()
{
    // according to the Meffert solution, 4 and 5 will have
    // the correct colors but might be oriented incorrectly.
    // check that they have the correct colors.
    assert(fMega.EdgeHasColors(kSouthPoleFace, 8, 1, 2));
    assert(fMega.EdgeHasColors(kSouthPoleFace, 9, 1, 3));
    // check that everything is correctly oriented
    bool b78OK = fMega.IsEdgeCorrect(kSouthPoleFace, 8);
    bool b79OK = fMega.IsEdgeCorrect(kSouthPoleFace, 9);

    if (b78OK && b79OK)
        return; // we're done!

    // if only one is bad, then the equator is also bad, so
    // loft it to the pole first
    fMega.WriteComment("Step8OrientFourFive lofting");
    enum Lofting {eNothing = 1, eLLL, eRLL};
    Lofting whichLoft = eNothing;
    if (b78OK && !b79OK)
    {
        whichLoft = eLLL; // loft to left
        DoLLL(8, 9);
    }
    else if (!b78OK && b79OK)
    {
        whichLoft = eRLL; // loft to right;
        DoRLL(8, 9);
    }

    // now the mis-oriented edges are on the South Pole;
    // apply the special operation to re-orient them
    fMega.WriteComment("Step8OrientFourFive placing");
    for (int i = 1; i <= 4; i++)
    {
        DoRLL(8, 9);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
    }
    fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
    for (int i = 1; i <= 4; i++)
    {
        DoRLL(8, 9);
        fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
    }
    fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);

    // now return the equatorial edge if needed
    // NOTE: we do the operation twice;
    // the published Meffert solution incorrectly shows it
    // only once.
    fMega.WriteComment("Step8OrientFourFive returning
equator");
    if (whichLoft == eLLL)
    {
        DoLLL(8, 9);
        DoLLL(8, 9);
    }
    else if (whichLoft == eRLL)
    {
        DoRLL(8, 9);
        DoRLL(8, 9);
    }
}

////////////////////////////////////
// Step 9. Placing the southern equatorial corners
////////////////////////////////////
//
// We swap corners to get the southern equatorial corners correctly
// placed. Our basic case is when one corner is on the South Pole
// and one is on the South Equator; we convert the other case
// (both on South Equator) to the first case by lofting one of the
// corners to the South Pole.
void CSolver::Step9()
{
    for (CMegaminx::vertex_t dst =
        CMegaminx::kFirstSouthEqVertex;
        dst <= CMegaminx::kLastSouthEqVertex; dst++)
    {
        if (fMega.IsCornerCorrectlyPlaced(dst))
            continue; // already OK, so skip

        // find src, the vertex holding the corner that should be here
        // src is either on the South Pole or on the Southern Equator
        //
        CMegaminx::color_t c0 =
            fMega.CorrectColor(CMegaminx::kCornerFaces[dst][0]);
        CMegaminx::color_t c1 =
            fMega.CorrectColor(CMegaminx::kCornerFaces[dst][1]);
        CMegaminx::color_t c2 =
            fMega.CorrectColor(CMegaminx::kCornerFaces[dst][2]);
        CMegaminx::vertex_t src =
            fMega.CornerHavingColors(c0, c1, c2);
        if (src <= CMegaminx::kLastSouthEqVertex)
        {
            // src is on South Equator, so we must loft it

```

A New System Has Arrived.



**Don't Get Caught
With Your
Mac Down!**



No Mac is complete without Timbuktu Pro,
the premier remote control and file transfer
software for Mac OS for over ten years.

No network is complete without the smart
systems management of netOctopus.

Both tools are newly rewritten for OS X,
bringing the power and simplicity of Netopia
software to the world's most advanced
operating system. For deployment, training
and support of all your Macs (and PC's!)
Timbuktu and netOctopus are indispensable.

Are you ready to migrate? We're ready to take you there.

macosxready.com

Timbuktu Pro

netopia

netOctopus

```

// to the South Pole
fMega.WriteComment("Step 9 lofting");
CMegaminx::face_t leftFace =
    CMegaminx::kCornerFaces[src][2];
CMegaminx::face_t rightFace =
    CMegaminx::kCornerFaces[src][1];
DoRLL(leftFace, rightFace);
DoRLL(leftFace, rightFace);
DoRLL(leftFace, rightFace);
src += 5; // src is now on the South Pole
}
Step9EquatorAndPole(dst, src);
}

Step9Verify();

void CSolver::Step9EquatorAndPole(CMegaminx::vertex_t dst,
    CMegaminx::vertex_t src)
{
    // rotate the South Pole CCW so src is directly above dst;
    // we need to rotate it back when we are finished to avoid disturbing
    // the South Pole edges.
    int dist = Distance(dst + 5, src);
    fMega.WriteComment("Step9EquatorAndPole rotating pole");
    CTempRotate rotl(fMega, kSouthPoleFace, dist,
        CMegaminx::eCounterCW);

    // now swap the vertices
    fMega.WriteComment("Step9EquatorAndPole swapping");
    CMegaminx::face_t leftFace =
        CMegaminx::kCornerFaces[dst][2];
    CMegaminx::face_t rightFace =
        CMegaminx::kCornerFaces[dst][1];
    DoRLL(leftFace, rightFace);
    DoRLL(leftFace, rightFace);
    DoRLL(leftFace, rightFace);
}

/////////////////////////////////////////////////////////////////
// Step 10. Placement Of the South Pole corners
/////////////////////////////////////////////////////////////////
//
void CSolver::Step10()
{
    for (CMegaminx::vertex_t destCorner =
        CMegaminx::kFirstSouthPoleVertex;
        destCorner <= CMegaminx::kLastSouthPoleVertex;
        destCorner++)
    {
        if (fMega.IsCornerCorrectlyPlaced(destCorner))
            continue;

        // find the corner that belongs here; do not check
        // already-placed corners. Do not check the srcCorner
        // because we already know it is not correctly placed
        // (we don't do orientation until Step 11).
        bool bFound = false;
        for (CMegaminx::vertex_t srcCorner = destCorner + 1;
            srcCorner <= CMegaminx::kLastSouthPoleVertex &&
            !bFound;
            srcCorner++)
        {
            if (destCorner ==
                fMega.CorrectSouthernVertex(srcCorner))
            {
                bFound = true;
                // now move everybody; we always rotate to vertex 15,
                // and the left and right faces are 12 and 8.
                // We use two blocks so the CTempRotate destructors will
                // rotate back to the original position in between
                // transformations
                fMega.WriteComment("Step10");
                {
                    // swap srcCorner and 10
                    CTempRotate rotl(fMega, kSouthPoleFace,
                        srcCorner - 15, CMegaminx::eCounterCW);
                    DoRUU(12, 8);
                    DoRUU(12, 8);
                    DoRUU(12, 8);
                }

                // swap destCorner and srcCorner (which is now in 10)
            }
        }
    }
}

```

```

CTempRotate rot2(fMega, kSouthPoleFace,
    destCorner - 15, CMegaminx::eCounterCW);
DoRUU(12, 8);
DoRUU(12, 8);
DoRUU(12, 8);
}

{
    // restore 10 to its original position
    CTempRotate rotl(fMega, kSouthPoleFace,
        srcCorner - 15, CMegaminx::eCounterCW);
    DoRUU(12, 8);
    DoRUU(12, 8);
    DoRUU(12, 8);
}

}
assert(bFound);
}
Step10Verify();
}

/////////////////////////////////////////////////////////////////
// Step 11. Orientation Of the southern equatorial and South Pole corners
/////////////////////////////////////////////////////////////////
//
// We find pairs of oppositely-oriented corner pieces that are not
// correctly oriented, drop them to the South Pole, then
// swap them and return them to their original position. They
// have to be dropped such that they are next to each other.
// We treat the case "neither on South Pole" as the basic case
// and transform all others to that:
// (1) if both are on the South Pole, we pick two separate faces,
//     one holding each corner, and rotate those CCW to drop them
//     to the Southern Equatorial belt.
// (2) if one is on the South Pole and one not, we rotate the
//     South Pole so that corner is not touched the face the other is
//     on, then rotate a face the South Pole corner is on.

class CStep11CornerVisitor : public CCornerVisitor
{
public:
    CStep11CornerVisitor(CMegaminx& rMega) :
        fMega(rMega),
        fNeedsCounterCWCORner1(-1), fNeedsCounterCWCORner2(-1),
        fNeedsCWCORner1(-1), fNeedsCWCORner2(-1)
    {}
    ~CStep11CornerVisitor() {}

    virtual void VisitCorner(int cornerIndex):

        // member variables - these are the vertex indices
        // of corners that need 1 turn CCW or CW to be
        // correctly oriented
        CMegaminx::vertex_t fNeedsCounterCWCORner1;
        CMegaminx::vertex_t fNeedsCounterCWCORner2;
        CMegaminx::vertex_t fNeedsCWCORner1;
        CMegaminx::vertex_t fNeedsCWCORner2;

        CMegaminx& fMega;
};

void CStep11CornerVisitor::VisitCorner(int cornerIndex)
{
    // maybe we are already done
    if ((fNeedsCounterCWCORner1 >= 0) &&
        (fNeedsCWCORner1 >= 0))
        return;

    // check whether the corner is correctly oriented,
    // and if not, which direction it should be turned

    // face numbers in CCW order
    CMegaminx::face_t f0 =
        CMegaminx::kCornerFaces[cornerIndex][0];
    CMegaminx::face_t f1 =
        CMegaminx::kCornerFaces[cornerIndex][1];
    CMegaminx::face_t f2 =
        CMegaminx::kCornerFaces[cornerIndex][2];

    // facelet color for facelet on face f0
    CMegaminx::color_t c0 = fMega.CORnerFaceletColor(f0, f1,
        f2);
}

```

```

if (c0 == CMegaminx::CorrectColor(f1))
{
    // should turn CCW
    if (fNeedsCounterCWCORner1 < 0)
        fNeedsCounterCWCORner1 = cornerIndex;
    else if (fNeedsCounterCWCORner2 < 0)
        fNeedsCounterCWCORner2 = cornerIndex;
}
else if (c0 == CMegaminx::CorrectColor(f2))
{
    // should turn CW
    if (fNeedsCWCORner1 < 0)
        fNeedsCWCORner1 = cornerIndex;
    else if (fNeedsCWCORner2 < 0)
        fNeedsCWCORner2 = cornerIndex;
}
// otherwise is correctly oriented, do nothing
}

void CSolver::Step11()
{
    // the transformation turns one corner CW and one corner CounterCW,
    // so ideally we would pick corners that need this to be correctly
    // positioned; however if we don't have such a pair we can pick
    // two with the same positioning, and then one will become correctly
    // positioned and one will be switched to the opposite positioning.
    for (int i = 0; i < 100; i++) // break out if stuck in loop
    {
        CStep11CornerVisitor aVisitor(fMega);
        VisitAllCorners(aVisitor);
        int vCounterCW = aVisitor.fNeedsCounterCWCORner1;
        int vCW = aVisitor.fNeedsCWCORner1;
        if ((vCounterCW < 0) && (vCW < 0))
            break;

        // check that we the ideal pairing, and if not double up
        // on the other orientation
        if (vCounterCW < 0)
            vCounterCW = aVisitor.fNeedsCWCORner2;
        else if (vCW < 0)
            vCW = aVisitor.fNeedsCounterCWCORner2;
        assert(vCW >= 0 && vCounterCW >= 0);

        bool bCounterCWIsOnEquator =
            fMega.IsSouthEquatorVertex(vCounterCW);
        bool bCWIsOnEquator = fMega.IsSouthEquatorVertex(vCW);

        if (bCounterCWIsOnEquator && bCWIsOnEquator)
        {
            Step11BothEquators(vCounterCW, vCW);
        }
        else
        {
            // pick two non-adjacent faces for dropping the vertices
            // to the South Equator. If one is already on the equator
            // we don't have to move it. If the vertices are directly
            // above each other (one on pole and one on equator), we need
            // to rotate the South Pole first so they can be on
            // non-adjacent faces.

            // first check for possibly needed pole rotation
            int spCount = 0;
            if (std::abs(vCounterCW - vCW) == 5)
            {
                // the vertices are above each other, so we'll
                // rotate the pole 1 CCW
                spCount = 1;
                if (!bCounterCWIsOnEquator)
                    vCounterCW =
                        fMega.NextCounterCWVertex(kSouthPoleFace,
                                                  vCounterCW);
                else
                    vCW = fMega.NextCounterCWVertex(kSouthPoleFace,
                                                      vCW);
            }

            // now do any necessary dropping of vertices to the South Equator

            // check counterCW vertex
            int faceCounterCW = 0, faceCW = 0;
            fMega.FindNonAdjacentSouthFaces(vCounterCW, vCW,
                                             &faceCounterCW, &faceCW);

            int counterCWCount = 0, cwCount = 0;
            int nextCounterCW = vCounterCW, nextCW = vCW;

```

Take



With JobOrder

Estimating ■ Scheduling ■ Job Costing ■ Traffic ■ Billing
Reporting ■ Notifications ■ Accounting ■ Calendars

Discover the Affordable Management Software

free demo at www.JobOrder.com

toll-free 877.714.2587 phone 607.756.4150

JobOrderTM
maximizing business productivity

Macworld

Conference & Expo™



Register online www.macworldexpo.com



Conferences **July 15-19, 2002**
Expo **July 17-19, 2002**

Jacob K. Javits Center **New York**



For more information, call toll free
1-800-645-EXPO



 **IDG**
WORLD EXPO

© 2002 IDG World Expo. All rights reserved.
All other trademarks contained herein
are the property of the respective owners.

See hundreds of companies and thousands of products at the **largest technology show in New York.**

Macworld Conference & Expo is more than a conference, and more than an exposition. It's a **MUST ATTEND** staple for the Mac community. Enhance your knowledge, network with peers, personally interact with new products and technologies, and finalize your purchase decisions under one roof.

Personalize your educational experience, by mixing-and-matching conference programs.



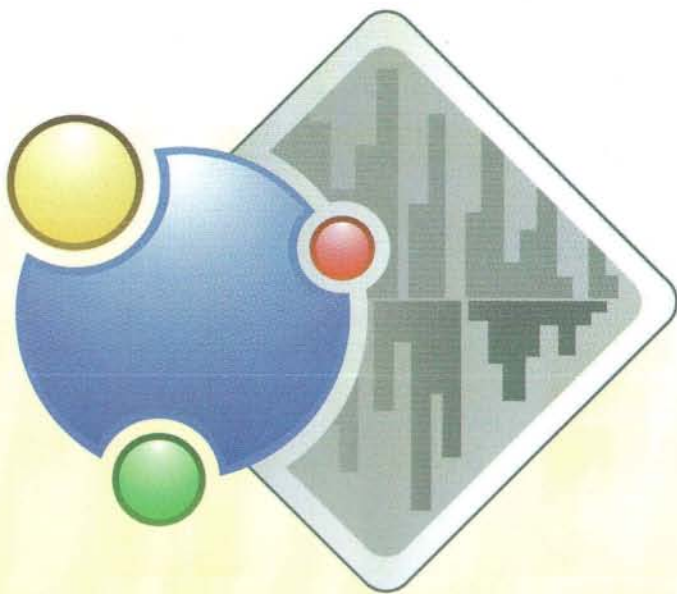
- **Macworld/Users** – Learn tips and tricks for your favorite application, how to maximize your digital capabilities, get a taste of Mac OS X.
- **Macworld/Power Tools** – Immerse yourself for two days with one of the most popular productivity tools for the Mac, and take your skill set to its top-level.
- **Macworld/Pro** – Participate in sophisticated training for Mac networking, digital video and filmmaking, professional publishing, Mac systems administrations and management, and detailed technical presentations that take you inside Mac OS X.
- **MacBeginnings** – Enjoy educational sessions full of tips, techniques and fact-filled training. Learn Mac basics, or about the Internet. Learn how to set-up and create desktop movies, or how to join and utilize a Macintosh user group. These sessions are open to all registered attendees.
- **Workshops** – Make the most out of your show experience by adding a full-day workshop to your educational agenda.
- **Brand New, Hands-on MacLabs** – Provide hands-on computer training on key applications and tools. Our trainers are experts in their field, and they are prepared to share their knowledge with you so select a discipline to focus on and bring your laptop!



Register online today using
Priority Code: **A-MTJ**
www.macworldexpo.com

Flagship Sponsors

Macworld Macworld.com  **MacCentral**



IPNetMonitorX

- The Swiss Army Knife of Mac OS X Internet Diagnostic Tools
- 15 Fully Integrated Tools—Monitor, Link Rate, Address Scan, TCP Dump and more
- Troubleshoot and Solve Network Problems
- Responsive, Intuitive, Easy to Use Interface
- Fast Multi-Threaded Architecture—see network behavior as it happens
- 21-Day Fully Functional Free Trial

**Only
\$40!**

Download Your Copy Now at

www.sustworks.com

SUSTAINABLE

Softw*orks*

Tools for Internet Travel



```
CMegaminx::Direction directionCounterCW =
CMegaminx::eCW,
    directionCW = CMegaminx::eCW;
    if (!bCounterCWIsOnEquator)
    {
        counterCWCount = 1;
        nextCounterCW = fMega.NextCWVertex(faceCounterCW,
            vCounterCW);
        directionCounterCW = CMegaminx::eCW;
        if (!fMega.IsSouthEquatorVertex(nextCounterCW))
        {
            // wrong direction, go in other direction
            nextCounterCW =
fMega.NextCounterCWVertex(faceCounterCW,
            vCounterCW);
            directionCounterCW = CMegaminx::eCounterCW;
        }
    }

    // check CW vertex
    if (!bCWIsOnEquator)
    {
        cwCount = 1;
        nextCW = fMega.NextCWVertex(faceCW, vCW);
        directionCW = CMegaminx::eCW;
        if (!fMega.IsSouthEquatorVertex(nextCW))
        {
            // wrong direction, go in other direction
            nextCW = fMega.NextCounterCWVertex(faceCW, vCW);
            directionCW = CMegaminx::eCounterCW;
        }
    }

    fMega.WriteComment("Step11 non-equator case");
    CTempRotate rotSouthPole(fMega, kSouthPoleFace,
spCount,
        CMegaminx::eCounterCW);
    CTempRotate rotCounterCW(fMega, faceCounterCW,
        counterCWCount, directionCounterCW);
    CTempRotate rotCW(fMega, faceCW, cwCount,
directionCW);
    Step11BothEquators(nextCounterCW, nextCW);
}

void CSolver::Step11BothEquators(int vCounterCW, int vCW)
{
    // we will loft the colors of both vertices to the South Pole;
    // need to figure out which direction to rotate their faces,
    // and what rotation is needed for the South Pole to have the
    // lofted corners together.

    // pick two non-adjacent faces for lofting the vertices
    int faceCounterCW, faceCW; // faces to rotate
    fMega.FindNonAdjacentSouthFaces(vCounterCW, vCW,
        &faceCounterCW, &faceCW);

    // figure out the direction to rotate each face, and which vertex
    // the corner will loft to
    // We will position the CCW corner 1 vertex CW of the CW face,
    // and rotate the CW face to put the CW vertex next to the CCW
    // vertex. The right face will then be the CW face.
    CMegaminx::Direction dCounterCW, dCW; // directions to loft
    int loftedCounterCW1, loftedCW1;

    loftedCounterCW1 = fMega.NextCounterCWVertex(faceCounterCW,
        vCounterCW);
    if (fMega.IsSouthPoleVertex(loftedCounterCW1))
    {
        dCounterCW = CMegaminx::eCounterCW;
    }
    else
    {
        // wrong direction, go back in other direction
        dCounterCW = CMegaminx::eCW;
        loftedCounterCW1 = fMega.NextCWVertex(faceCounterCW,
            vCounterCW);
    }
    int cwClicks = 0;
    loftedCW1 = fMega.NextCWVertex(faceCW, vCW);
    if (fMega.IsSouthPoleVertex(loftedCW1))
    {
```

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."

— MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

— MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

— Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"

— Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

— Bill Goodman, author of *Smaller Installer* and *Compact Pro*

"Resorcerer Rocks! Buy it, you will NOT regret it."

— Joe Zobkiw, author of *A Fragment of Your Imagination*

"Resorcerer will pay for itself many times over in saved time and effort."

— MacUser review

"The template that disassembles 'PICT's is awesome!"

— Bill Steinberg, author of *Pyro!* and *PBTools*

"Resorcerer proved indispensable in its own creation!"

— Doug McKenna, author of *Resorcerer*



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

Website price: \$128 - \$256

(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New
in
2.0:

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TEMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

```

    dCW = CMegaminx::eCW; // OK, at left edge of face
    cwClicks = 1;
}
else
{
    // wrong direction, go two vertices in other direction
    dCW = CMegaminx::eCounterCW;
    loftedCW1 = fMega.NextCounterCWVertex(faceCW, vCW);
    loftedCW1 = fMega.NextCounterCWVertex(faceCW, loftedCW1);
    cwClicks = 2;
}

// now figure out the South Pole rotation
// we want CCW to be on the left of CW
// as a simplification we will always rotate the South Pole CCW
int iCounterCW = -1, iCW = -1;
for (int i = 0; i < 5; i++)
{
    if (CMegaminx::kFaceVertices[kSouthPoleFace][i] ==
        loftedCounterCW1)
        iCounterCW = i;
    else if (CMegaminx::kFaceVertices[kSouthPoleFace][i] ==
        loftedCW1)
        iCW = i;
}
int distToRotate = (iCW - 1) - iCounterCW;
if (distToRotate < 0)
    distToRotate += 5;
if (distToRotate >= 5)
    distToRotate -= 5;

// OK, now we are ready to rotate everything!
int rightFace = faceCW;
int leftFace = faceCW - 1;
if (leftFace <= kSouthPoleFace)
    leftFace += 5;

// rotate the corners into place
fMega.WriteComment("Step11BothEquators lofting");
CTempRotate loftCounterCW(fMega, faceCounterCW, 1,
    dCounterCW);
CTempRotate southPoleRotate(fMega, kSouthPoleFace,
    distToRotate, CMegaminx::eCounterCW);
CTempRotate loftCW(fMega, faceCW, cwClicks, dCW);

// do the corner swap
fMega.WriteComment("Step11BothEquators corner swap");
DoRUU(leftFace, rightFace);
DoRUU(leftFace, rightFace);
fMega.Slice(kSouthPoleFace, CMegaminx::eCounterCW, 1);
DoLUU(leftFace, rightFace);
DoLUU(leftFace, rightFace);
fMega.Slice(kSouthPoleFace, CMegaminx::eCW, 1);
}

#pragma mark === Verification Routines ===
//Verification Routines
// see online code archive
CMegaminxApp.cpp
// see online code archive
CMegaminx.cpp
#include "CMegaminx.h"
#include "CMegaminxApp.h"

#include <cassert>

// initialization of tables

const CMegaminx::vertex_t CMegaminx::kFaceVertices[13][5] =
{
    // dummy face for 0
    {0, 0, 0, 0, 0},

    // North Pole face
    {0, 1, 2, 3, 4},

    // Northern Equatorial faces
    {3, 2, 7, 12, 8},
    {2, 1, 6, 11, 7},
    {1, 0, 5, 10, 6},
    {0, 4, 9, 14, 5},
    {4, 3, 8, 13, 9},

    // South Pole face
    {19, 18, 17, 16, 15},

    // Southern Equatorial faces
    {19, 15, 10, 5, 14},
    {18, 19, 14, 9, 13},
    {17, 18, 13, 8, 12},
    {16, 17, 12, 7, 11},
    {15, 16, 11, 6, 10}
};

const CMegaminx::face_t CMegaminx::kCornerFaces[20][3] =
{
    // North Pole
    {1, 5, 4},
    {1, 4, 3},
    {1, 3, 2},
    {1, 2, 6},
    {1, 6, 5},

    // Northern Equatorial
    {4, 5, 8},
    {3, 4, 12},
    {2, 3, 11},
    {2, 10, 6},
    {5, 6, 9},

    // Southern Equatorial
    {4, 8, 12},
    {3, 12, 11},
    {2, 11, 10},
    {6, 10, 9},
    {5, 9, 8},

    // South Pole
    {7, 12, 8},
    {7, 11, 12},
    {7, 10, 11},
    {7, 9, 10},
    {7, 8, 9}
};

// list of adjacent face numbers, indexed by face number.
// each item lists the faces adjacent to this one,
// in counterclockwise order as viewed from above this face.
// This list must be coordinated with the vertex list so that
// face[1] touches vertices [0] and [1].
const CMegaminx::face_t CMegaminx::kAdjacentFaces[13][5] =
{
    {0, 0, 0, 0, 0}, // dummy for face 0

    {4, 3, 2, 6, 5},
    {1, 3, 11, 10, 6},
    {1, 4, 12, 11, 2},
    {1, 5, 8, 12, 3},
    {1, 6, 9, 8, 4},
    {1, 2, 10, 9, 5},

    {9, 10, 11, 12, 8},
    {7, 12, 4, 5, 9},
    {7, 8, 5, 6, 10},
    {7, 9, 6, 2, 11},
    {7, 10, 2, 3, 12},
    {7, 11, 3, 4, 8}
};

const CMegaminx::face_t CMegaminx::kFaceBelow[20] =
{5, 4, 3, 2, 6, 8, 12, 11, 10, 9, 8, 12, 11, 10, 9, 0, 0, 0,
0, 0};
const CMegaminx::face_t CMegaminx::kFaceAbove[20] =
{0, 0, 0, 0, 0, 4, 3, 2, 6, 5, 4, 3, 2, 6, 5, 12, 11, 10, 9,
8};

const CMegaminx::face_t CMegaminx::kFaceDownRight[13] =
{0, 0, 10, 11, 12, 8, 9, 0, 0, 0, 0, 0, 0};
const CMegaminx::face_t CMegaminx::kFaceDownLeft[13] =
{0, 0, 11, 12, 8, 9, 10, 0, 0, 0, 0, 0, 0};
const CMegaminx::face_t CMegaminx::kFaceUpRight[13] =
{0, 0, 0, 0, 0, 0, 0, 0, 4, 5, 6, 2, 3};

```

You'll be stuck in New York City's heaviest traffic and loving every minute of it.

At Macworld Conference & Expo NY 2002, we make it easier for all techies, both novice and expert, to have a space as special as their ideas. **MacTech® Magazine** presents **MacTech_(sm) Central** — at MWNY 2002 showcasing technical and developer tool/service companies for the Macintosh in the biggest possible way. Macworld Conference & Expo takes place in New York City, July 15-19, 2002. Check out the exhibits (July 17-19) at the show and on the web at: www.mactechcentral.com.



MacTech is a registered trademark of Xplain Corporation. MacTech Central is a service mark of Xplain Corporation.
Other trademarks and copyrights appearing in this printing remain the property of their respective holders.

```
const CMegaminx::face_t CMegaminx::kFaceUpLeft[13] =
{0, 0, 0, 0, 0, 0, 0, 0, 5, 6, 2, 3, 4};
```

// list of North Pole edges

```
const CMegaminx::face_t
CMegaminx::kNorthPoleEdgeN[kNumNorthPoleEdges] =
{1, 1, 1, 1, 1};
const CMegaminx::face_t
CMegaminx::kNorthPoleEdgeS[kNumNorthPoleEdges] =
{2, 3, 4, 5, 6};
```

// list of North Equator edges.

```
const CMegaminx::face_t
CMegaminx::kNorthEqEdgeL[kNumNorthEqEdges] =
{2, 3, 4, 5, 6};
const CMegaminx::face_t
CMegaminx::kNorthEqEdgeR[kNumNorthEqEdges] =
{6, 2, 3, 4, 5};
```

// list of middle equatorial edges. Ordered in two arrays,
// the first giving the south face and the second the corresponding
// north face. For even indices the edge is below and right of the
// north face, for odd indices it is below and to the left.

```
const CMegaminx::face_t
CMegaminx::kMiddleEqEdgeN[kNumMiddleEqEdges] =
{5, 4, 4, 3, 3, 2, 2, 6, 6, 5};
const CMegaminx::face_t
CMegaminx::kMiddleEqEdgeS[kNumMiddleEqEdges] =
{8, 8, 12, 12, 11, 11, 10, 10, 9, 9};
```

// list of Southern Equator edges

```
const CMegaminx::face_t
CMegaminx::kSouthEqEdgeL[kNumSouthEqEdges] =
{8, 9, 10, 11, 12};
const CMegaminx::face_t
CMegaminx::kSouthEqEdgeR[kNumSouthEqEdges] =
{12, 8, 9, 10, 11};
```

// list of South Pole edges

```
const CMegaminx::face_t
CMegaminx::kSouthPoleEdgeN[kNumSouthPoleEdges] =
{8, 9, 10, 11, 12};
const CMegaminx::face_t
CMegaminx::kSouthPoleEdgeS[kNumSouthPoleEdges] =
{7, 7, 7, 7, 7};
```

////////////////////////////////////

```
CMegaminx::CMegaminx(const string& testNumberString)
```

```
{
    // clear all facelets
    std::memset(fEdgeFacelets, 0, sizeof(fEdgeFacelets));
    std::memset(fCornerFacelets, 0, sizeof(fCornerFacelets));
```

// open correct rotations file

```
string rotationsName = string("rotations") +
    testNumberString + ".txt";
fRotationsStream.open(rotationsName.c_str());
if (!fRotationsStream.is_open())
{
    CMegaminxApp::SayFileError(rotationsName);
    return;
}
```

```
CMegaminx::~CMegaminx()
```

```
{
    fRotationsStream.close();
}
```

```
void CMegaminx::LoadCornerFacelet(face_t faceNumber1, face_t
faceNumber2,
```

```
    face_t faceNumber3, color_t colorNumber)
{
    assert(1 <= faceNumber1 && faceNumber1 <= 12);
    assert(1 <= faceNumber2 && faceNumber2 <= 12);
    assert(1 <= faceNumber3 && faceNumber3 <= 12);

    if (faceNumber2 < faceNumber3)
        fCornerFacelets[faceNumber1][faceNumber2][faceNumber3]
            = colorNumber;
    else
        fCornerFacelets[faceNumber1][faceNumber3][faceNumber2]
            = colorNumber;
```

```
}

void CMegaminx::LoadEdgeFacelet(face_t faceNumber1, face_t
faceNumber2,
```

```
    color_t colorNumber)
{
    assert(1 <= faceNumber1 && faceNumber1 <= 12);
    assert(1 <= faceNumber2 && faceNumber2 <= 12);

    fEdgeFacelets[faceNumber1][faceNumber2] = colorNumber;
}
```

```
void CMegaminx::SliceImp(CMegaminx::face_t faceNumber,
```

```
    Direction direction)
{
    short *pFaceletColors1[5], *pFaceletColors2[5],
        *pFaceletColors3[5]; // pointers to colors to move, listed
    CCW
    int i;
```

// rotate the edge facelets

```
for (i = 0; i < 5; i++)
{
    int adj = kAdjacentFaces[faceNumber][i];
    pFaceletColors1[i] =
&fEdgeFacelets[adj][faceNumber];
    // adjacent face
    pFaceletColors2[i] =
&fEdgeFacelets[faceNumber][adj];
    // this face
    RotateFacelets(pFaceletColors1, direction);
    RotateFacelets(pFaceletColors2, direction);
}
```

// rotate the corner facelets

```
for (i = 0; i < 5; i++)
{
    int adjRight =
        kAdjacentFaces[faceNumber][(i == 0) ? 4 : i -
1];
    int adjLeft = kAdjacentFaces[faceNumber][i];
    int low, high;

    low = (adjRight < adjLeft) ? adjRight : adjLeft;
    high = (adjRight > adjLeft) ? adjRight : adjLeft;
    pFaceletColors1[i] =
        &fCornerFacelets[faceNumber][low][high]; // this face

    low = (faceNumber < adjLeft) ? faceNumber : adjLeft;
    high = (faceNumber > adjLeft) ? faceNumber :
adjLeft;
    pFaceletColors2[i] =
        &fCornerFacelets[adjRight][low][high]; // right
face
    low = (faceNumber < adjRight) ? faceNumber :
adjRight;
    high = (faceNumber > adjRight) ? faceNumber :
adjRight;
    pFaceletColors3[i] =
        &fCornerFacelets[adjLeft][low][high]; // left face
    RotateFacelets(pFaceletColors1, direction);
    RotateFacelets(pFaceletColors2, direction);
    RotateFacelets(pFaceletColors3, direction);
}
```

// write out this rotation to the file

```
fRotationsStream << faceNumber << ".";
fRotationsStream << ((direction == eCW) ? '+' : '-');
fRotationsStream << std::endl;
```

```
void CMegaminx::Slice(face_t faceNumber, Direction direction,
```

```
    int clicks)
{
    assert(clicks >= 0);
    for (int i = 0; i < clicks; i++)
        SliceImp(faceNumber, direction);
}
```

```
bool CMegaminx::IsSolved()
```

```
{
    bool bSolved = true;
```

```

for (int i = 1; i <= 12; i++)
{
    int rightColor = CorrectColor(i);
    for (int j = 1; j <= 12; j++)
    {
        // check edges
        bSolved = bSolved && (fEdgeFacelets[i][j] == 0 ||
            fEdgeFacelets[i][j] == rightColor);

        // check corners
        for (int k = j + 1; k <= 12; k++)
        {
            bSolved = bSolved && (fCornerFacelets[i][j][k] == 0 ||
                fCornerFacelets[i][j][k] == rightColor);
        }
    }
    return bSolved;
}

void CMegaminx::RotateFacelets(short **ppColor, Direction
direction)
{
    // rotate a list of 5 facelet colors
    // the pList is an array of 5 pointer to the color entries
    // in either fEdgeFacelets or fCornerFacelets, in counterclockwise
    // order. For CCW direction we shift the array right, and
    // for CW direction we shift it left.
    short saveColor = 0;
    int i;
    if (direction == eCounterCW)
    {
        // shift right
        saveColor = *(ppColor + 4);
        for (i = 3; i >= 0; i--)
            *(ppColor + i + 1) = *(ppColor + i);
        *(ppColor + 0) = saveColor;
    }
    else
    {
        // shift left

```

```

        saveColor = *(ppColor + 0);
        for (i = 0; i < 4; i++)
            *(ppColor + i) = *(ppColor + i + 1);
        *(ppColor + 4) = saveColor;
    }
}

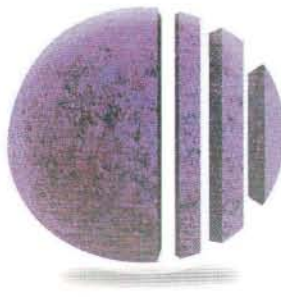
bool CMegaminx::IsCornerCorrectlyPlaced(vertex_t vertex)
{
    // get the actual colors and the correct colors;
    // the item is correctly placed if these lists are
    // rotations of each other.
    int f0, f1, f2;
    int actualColors[5];
    int correctColors[3];

    f0 = kCornerFaces[vertex][0];
    f1 = kCornerFaces[vertex][1];
    f2 = kCornerFaces[vertex][2];
    actualColors[0] = actualColors[3] =
        CornerFaceletColor(f0, f1, f2);
    actualColors[1] = actualColors[4] =
        CornerFaceletColor(f1, f2, f0);
    actualColors[2] = CornerFaceletColor(f2, f0, f1);
    correctColors[0] = kCornerFaces[vertex][0];
    correctColors[1] = kCornerFaces[vertex][1];
    correctColors[2] = kCornerFaces[vertex][2];
    if (correctColors[0] > 6)
        correctColors[0] -= 6;
    if (correctColors[1] > 6)
        correctColors[1] -= 6;
    if (correctColors[2] > 6)
        correctColors[2] -= 6;

    bool bHaveMatch = false;
    for (int i = 0; (i < 3) && !bHaveMatch; i++)
    {
        bHaveMatch = true;
        for (int j = 0; j < 3; j++)
        {
            if (actualColors[i+j] != correctColors[j])
                bHaveMatch = false;
        }
    }
}

```

YOU'VE GOT YOUR NEW MAC. GREAT. NOW, ALL YOU WANT TO DO IS PUBLISH A SIMPLE, YET ELEGANT WEBSITE, CATALOG AND POST YOUR PICTURES, DO SOME PUBLISHING, DO A LITTLE DRAWING, MAKE A FEW PDF FILES, ARCHIVE SOME DOCUMENTS TO SEND TO YOUR MOTHER, CUT UP SOME BUTTONS FOR THAT ELEGANT WEBSITE, ANIMATE THEM, & TRACK HOW MUCH TIME IT TOOK YOU TO DO IT ALL. AND YOU WANT TO DO IT FOR LESS THAN \$300.



STONE STUDIO™ \$299



DOWNLOAD NOW OR PURCHASE CD WWW.STONE.COM

STONE STUDIO™ 8 APPS THAT SET YOU FREE TO DRAW, TWEAK, PROCESS AND PUBLISH YOUR IDEAS. ON PAPER OR ON THE WEB.

```

    }
    return bHaveMatch;
}

bool CMegaminx::IsCornerCorrect(vertex_t vertex)
{
    int f0 = kCornerFaces[vertex][0];
    int f1 = kCornerFaces[vertex][1];
    int f2 = kCornerFaces[vertex][2];

    bool bOK =
        CornerFaceletColor(f0, f1, f2) == CorrectColor(f0)
    &&
        CornerFaceletColor(f1, f2, f0) == CorrectColor(f1)
    &&
        CornerFaceletColor(f2, f0, f1) == CorrectColor(f2);

    return bOK;
}

CMegaminx::vertex_t
CMegaminx::FacesToVertex(CMegaminx::face_t f0,
    CMegaminx::face_t f1, CMegaminx::face_t f2)
{
    // find the lowest face number, then search the table of corner faces
    // for a match. It is an error if we don't find a match.
    int holdFace = 0;

    if (f1 < f0)
    {
        holdFace = f0;
        f0 = f1;
        f1 = holdFace;
    }
    if (f2 < f0)
    {
        holdFace = f0;
        f0 = f2;
        f2 = holdFace;
    }

    for (int i = 0; i < 20; i++)
    {
        if (f0 == kCornerFaces[i][0])
        {
            int trialFace1 = kCornerFaces[i][1];
            int trialFace2 = kCornerFaces[i][2];
            if ( (f1 == trialFace1 && f2 == trialFace2) ||
                (f1 == trialFace2 && f2 == trialFace1) )
            {
                return i;
            }
        }
    }
}

```

```

    }
    // trouble, did not find a match
    ::SysBeep(1);
    assert(false);
    return 0;
}

CMegaminx::vertex_t
CMegaminx::CorrectSouthernVertex(CMegaminx::vertex_t vertex)
{
    // find where v1 should be;
    // first read out its current colors, then
    // figure its correct faces
    int oldf0, oldf1, oldf2; // old face numbers
    int c0, c1, c2; // corresponding current colors
    int newf0, newf1, newf2; // new face numbers
    oldf0 = kCornerFaces[vertex][0];
    oldf1 = kCornerFaces[vertex][1];
    oldf2 = kCornerFaces[vertex][2];
    c0 = CornerFaceletColor(oldf0, oldf1, oldf2);
    c1 = CornerFaceletColor(oldf1, oldf2, oldf0);
    c2 = CornerFaceletColor(oldf2, oldf0, oldf1);

    // in general we can find the face numbers by adding 6
    // to each color; however for Southern Equatorial
    // vertices there is one color that should not have 6
    // added, and that is the "non-contiguous" color.
    newf0 = c0 + 6;
    newf1 = c1 + 6;
    newf2 = c2 + 6;
    if (c0 != 1 && c1 != 1 && c2 != 1)
    {
        // not pole, so correct one face
        if (std::abs(newf0 - newf1) == 1 ||
            std::abs(newf0 - newf1) == 4)
            newf2 -= 6;
        else if (std::abs(newf1 - newf2) == 1 ||
            std::abs(newf1 - newf2) == 4)
            newf0 -= 6;
        else
            newf1 -= 6;
    }
    return FacesToVertex(newf0, newf1, newf2);
}

CMegaminx::vertex_t
CMegaminx::CornerHavingColors(CMegaminx::color_t c0,
    CMegaminx::color_t c1, CMegaminx::color_t c2)

```



REALbasic

AppleScript Director

Java SQL

C++ Revolution

XCMD MetaCard

SuperCard WebSiphon

futureBASIC VisualBasic

Valentina

Object-Relational SQL Database

**The fastest database engine
for MacOS/Windows**

**It operates 100's and sometimes
a 1000 times faster than other systems**

www.paradigmasoft.com

Hosted by macserve.net

Download full featured evaluation version

Introducing

**Collaborative Commerce
for Small-to-Midsize Businesses
and for People, too**

**For
Mac OS X
Linux
Windows**

MyBooks®
MyBooks Professional
Appgen® Custom Suite
Moneydance®

Fourth generation
development environment

Variable-length
relational database

Commerce modules

Source code

visit www.appgen.com or
call 1 800 231 0062



APPGEN®
Collaborative Commerce Platform

```

    int desiredColors[5];
    desiredColors[0] = desiredColors[3] = c0;
    desiredColors[1] = desiredColors[4] = c1;
    desiredColors[2] = c2;
    int trialVertex;
    for (trialVertex = 0; trialVertex < 20; trialVertex++)
    {
        // get the actual colors and the correct colors;
        // the item is correctly placed if these lists are
        // rotations of each other.
        int f0, f1, f2;
        f0 = kCornerFaces[trialVertex][0];
        f1 = kCornerFaces[trialVertex][1];
        f2 = kCornerFaces[trialVertex][2];
        int trialColors[3];
        trialColors[0] = CornerFaceletColor(f0, f1, f2);
        trialColors[1] = CornerFaceletColor(f1, f2, f0);
        trialColors[2] = CornerFaceletColor(f2, f0, f1);

        bool bHaveMatch = false;
        for (int i = 0; (i < 3) && !bHaveMatch; i++)
        {
            bHaveMatch = true;
            for (int j = 0; j < 3; j++)
            {
                if (desiredColors[i+j] != trialColors[j])
                    bHaveMatch = false;
            }

            if (bHaveMatch)
                break;
        }

        return trialVertex;
    }

CMegaminx::vertex_t
CMegaminx::NextCounterCWVertex(CMegaminx::face_t faceNumber,
    CMegaminx::vertex_t vertexNumber)
{
    for (int i = 0; i < 5; i++)
    {
        if (kFaceVertices[faceNumber][i] == vertexNumber)
            return kFaceVertices[faceNumber][(i == 4) ? 0 : i +
1];
    }

    ::SysBeep(1); // trouble, vertex not on face
    assert(false);
    return -1;
}

CMegaminx::vertex_t CMegaminx::NextCWVertex(CMegaminx::face_t
    faceNumber,
    CMegaminx::face_t vertexNumber)
{
    for (int i = 0; i < 5; i++)
    {
        if (kFaceVertices[faceNumber][i] == vertexNumber)
            return kFaceVertices[faceNumber][(i == 0) ? 4 : i -
1];
    }

    ::SysBeep(1); // trouble, vertex not on face
    assert(false);
    return -1;
}

void CMegaminx::FindNonAdjacentSouthFaces(CMegaminx::vertex_t
    v1,
    CMegaminx::vertex_t v2,
    CMegaminx::face_t *pf1, CMegaminx::face_t *pf2)
{
    int f1[2], f2[2]; // candidate faces

    f1[0] = kCornerFaces[v1][1];
    f1[1] = kCornerFaces[v1][2];
    f2[0] = kCornerFaces[v2][1];
    f2[1] = kCornerFaces[v2][2];

```

```

    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            int dist = f1[i] - f2[j];
            if (dist < 0)
                dist = -dist;
            if (dist == 2 || dist == 3)
            {
                *pf1 = f1[i];
                *pf2 = f2[j];
                return;
            }
        }
    }

    ::SysBeep(1); // trouble, couldn't find suitable faces
    assert(false);
}

#ifdef NDEBUG
void CMegaminx::WriteComment(const char *pComment)
{
    fRotationsStream << " " << pComment << std::endl;
}
#else
void CMegaminx::WriteComment(const char /* pComment */)
{
    // nothing
}
#endif

bool CMegaminx::EdgeHasColors(CMegaminx::face_t f0,
    CMegaminx::face_t f1,
    CMegaminx::color_t c0, CMegaminx::color_t c1)
{
    int actualc0 = fEdgeFacelets[f0][f1];
    int actualc1 = fEdgeFacelets[f1][f0];
    bool bMatches = ((actualc0 == c0) && (actualc1 == c1)) ||
        ((actualc0 == c1) && (actualc1 == c0));
    return bMatches;
}

#pragma mark === CTempRotate ===
////////////////////////////////////
CTempRotate::CTempRotate(CMegaminx& rMega, CMegaminx::face_t
    faceNumber,
    int clicks, CMegaminx::Direction direction) :
    fMega(rMega),
    fFaceNumber(faceNumber),
    fClicks(clicks),
    fDirection(direction)
{
    assert(fClicks >= 0);
    fMega.WriteComment("CTempRotate ctor");
    fMega.Slice(fFaceNumber, direction, fClicks);
}

CTempRotate::~CTempRotate()
{
    fMega.WriteComment("CTempRotate dtor");
    fMega.Slice(fFaceNumber,
        fDirection == CMegaminx::eCW ?
            CMegaminx::eCounterCW : CMegaminx::eCW,
        fClicks);
}

```

CMEGAMINXAPP.H

// see online code archive

CMEGAMINX.H

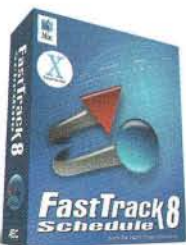
```

////////////////////////////////////
//
// The CMegaminx class holds the state of the Megaminx. There's only
// one operation for changing state, the Slice function. This class
// also handles writing out the rotations files; this placement is
// somewhat arbitrary, but is useful because it ensures that changing
// the Megaminx state will always cause the correct movements to be
// written out.
//

```

[illegible]

Journal of Interpersonal Violence 27(10) 1927-1941
© The Author(s) 2012
Reprints and permissions: sagepub.com/journalsPermissions.nav
DOI: 10.1177/0886260512468101
jiv.sagepub.com



Introducing FastTrack Schedule 8. Redesigned for Mac OS X and packed with new productivity features and a bold Aqua interface—FastTrack Schedule 8 has all the tools you need to ensure project success. For a free demo version or to order, call us today at 800.450.1983.



AEC
SOFTWARE

```

////////////////////////////////////
#pragma once

#include <fstream>
#include <string>
using std::string;

class CMegaminx
{
public:
    //////////////////////////////////////
    // various types and tables describing the Megaminx

    // enum for rotation direction; always measured looking
    // down on a face.
    // We also use this for an orientation of a corner; if
    // the color number increase CCW we call it CCW, and if
    // they increase CW we call it CW.
    enum Direction
    {
        eCounterCW = 1,
        eCW = 2
    };

    // typedefs for face number, color number, vertex number
    typedef int face_t;
    typedef int color_t;
    typedef int vertex_t;

    // vertices are numbered 0-19; these equates give the ranges
    static const int kNumVertices = 20;
    static const vertex_t kFirstNorthPoleVertex = 0;
    static const vertex_t kLastNorthPoleVertex = 4;
    static const vertex_t kFirstNorthEqVertex = 5;
    static const vertex_t kLastNorthEqVertex = 9;
    static const vertex_t kFirstSouthEqVertex = 10;
    static const vertex_t kLastSouthEqVertex = 14;
    static const vertex_t kFirstSouthPoleVertex = 15;
    static const vertex_t kLastSouthPoleVertex = 19;

    // vertex numbers of each face, indexed 0 through 19,
    // counterclockwise looking at the face from outside
    static const vertex_t kFaceVertices[13][5];

    // list of all vertices and the adjoining faces. Faces are listed
    // in CCW order started with the lowest-numbered.
    static const face_t kCornerFaces[20][3];

    // list of adjacent face numbers, indexed by face number.
    // each item lists the faces adjacent to this one,
    // in counterclockwise order as viewed from above this face.
    // This list must be coordinated with the vertex list so that

    // face[1] touches vertices [0] and [1].
    static const face_t kAdjacentFaces[13][5];

    // list of faces below or below left of a vertex, indexed
    // by vertex number. For North Equatorial vertices the face is
    // below (the vertex is its top vertex) and for North Pole
    // and South Equatorial vertices the face is below and
    // to the left (the vertex is its upper right vertex).
    static const face_t kFaceBelow[20];

    // list of faces above or above right of a vertex, indexed
    // by vertex number. For South Equatorial vertices the face is
    // above (the vertex is its bottom vertex) and for South Pole
    // and North Equatorial vertices the face is above and
    // to the right (the vertex is its lower left vertex).
    static const face_t kFaceAbove[20];

    // for equatorial faces, the faces above and below them.
    //
    // faces below and to left or right of given North Equatorial
    // face; indexed by face number
    static const face_t kFaceDownRight[13];
    static const face_t kFaceDownLeft[13];
    // faces above and to left or right of given South Equatorial
    // face; indexed by face number
    static const face_t kFaceUpRight[13];
    static const face_t kFaceUpLeft[13];

    // list of North Pole edges
    static const int kNumNorthPoleEdges = 5;
    static const face_t kNorthPoleEdgeN[kNumNorthPoleEdges];
    static const face_t kNorthPoleEdgeS[kNumNorthPoleEdges];

    // list of North Equator edges.
    static const face_t kNumNorthEqEdges = 5;
    static const face_t kNorthEqEdgeL[kNumNorthEqEdges];
    static const face_t kNorthEqEdgeR[kNumNorthEqEdges];

    // list of middle equatorial edges. Ordered in two arrays,
    // the first giving the north face and the second the corresponding
    // south face. For even indices the edge is below and right of the
    // north face, for odd indices it is below and to the left.
    static const int kNumMiddleEqEdges = 10;
    static const face_t kMiddleEqEdgeN[kNumMiddleEqEdges];
    static const face_t kMiddleEqEdgeS[kNumMiddleEqEdges];

    // list of Southern Equator edges
    static const int kNumSouthEqEdges = 5;
    static const face_t kSouthEqEdgeL[kNumSouthEqEdges];
    static const face_t kSouthEqEdgeR[kNumSouthEqEdges];

    // list of South Pole edges
    static const int kNumSouthPoleEdges = 5;

```

eat.
sleep.
Learn
Cocoa®.



Big
nerd
ranch

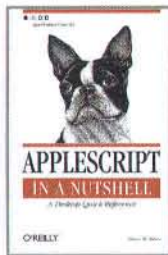
Intensive Classes for Programmers
www.bignerdranch.com
404.210.5663

Trying to keep up with OS X?

WE CAN GET YOU ON THE INSIDE TRACK.

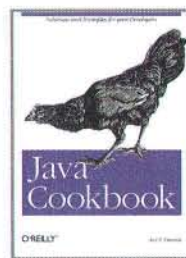
WE'RE TRUE INSIDERS.

O'Reilly and Apple have joined forces to bring you unparalleled access to the experts developing OS X. Get on the inside track (and stay there) with our OS X books and the O'Reilly Network's Mac DevCenter at www.oreillynet.com/mac.



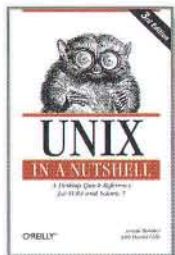
WE KEEP PACE WITH JAVA.

Our Java books—and our online site for Java developers, on.java.com—offer the most comprehensive and up-to-date information available to help you keep pace, too.



WE KNOW UNIX—INSIDE AND OUT.

We cut our teeth on Unix: O'Reilly has been publishing the most respected and useful books on Unix for well over a decade, and we're still at the head of the pack.



O'REILLY®

```

static const face_t kSouthPoleEdgeN[kNumSouthPoleEdges];
static const face_t kSouthPoleEdgeS[kNumSouthPoleEdges];

////////////////////////////////////////
// public functions

CMegaminx(const string& testNumberString);
~CMegaminx();

// load a corner facelet;
// faceNumber1 is the face it is on (numbered 1..12),
// faceNumber2 and faceNumber3 are neighboring faces,
// and colorNumber is the facelet's color (numbered 1..6).
void LoadCornerFacelet(face_t faceNumber1, face_t
faceNumber2,
        face_t faceNumber3, color_t colorNumber);

// similarly, load an edge facelet (no face 3 for these)
void LoadEdgeFacelet(face_t faceNumber1, face_t
faceNumber2,
        color_t colorNumber);

// slice operation; rotate face one click in given direction;
// changes our internal state and writes a line to
// the rotations file.
// This is a public function so that our helper classes
// can get to it.
void Slice(face_t faceNumber, Direction direction, int
clicks);

// check that the Megaminx is correctly solved
bool IsSolved();

// check whether a corner is correctly placed, based
// on its colors. The first checks only placement, not
// orientation.
bool IsCornerCorrectlyPlaced(vertex_t vertex);
bool IsCornerCorrect(vertex_t vertex);

// check whether an edge is correctly placed and positioned
bool IsEdgeCorrect(face_t faceL, face_t faceR)
{ return (fEdgeFacelets[faceL][faceR] ==
CorrectColor(faceL) &&
        fEdgeFacelets[faceR][faceL] ==
CorrectColor(faceR)); };

// look up the vertex having these faces
vertex_t FacesToVertex(face_t f0, face_t f1, face_t f2);

// find correct location of the colors at a vertex,
// assuming they should be in the Southern
// half. Returns the correct vertex number.
// We assume the colors actually belong in the
// specified Southern half, so caller
// must check this. "Southern" means South Pole
// or South Equatorial.
// Corners with same colors have opposite orientations
// in the Northern and Southern halves.
vertex_t CorrectSouthernVertex(vertex_t vertex);

// find the corner having these colors in this order
// (with rotations allowed). This means the corner that
// is currently colored with these corners.
color_t CornerHavingColors(color_t c0, color_t c1, color_t
c2);

// return facelet color of face f0 that borders f1 and f2
color_t CornerFaceletColor(face_t f0, face_t f1, face_t f2)
{ if (f1 < f2) return fCornerFacelets[f0][f1][f2];
  else return fCornerFacelets[f0][f2][f1]; };

// return color of edge on f0 that borders f1
color_t EdgeFaceletColor(face_t f0, face_t f1)
{ return fEdgeFacelets[f0][f1]; };

// return correct color of solved Megaminx face
static color_t CorrectColor(face_t faceNumber)
{ return (faceNumber <= 6) ? faceNumber : faceNumber -
6; };

// find next vertex on a face
static vertex_t NextCounterCWVertex(face_t faceNumber,
vertex_t vertexNumber);
static vertex_t NextCWVertex(face_t faceNumber, vertex_t
vertexNumber);

// find next or previous (numerically) South Equatorial faces
static face_t NextSouthEqFace(face_t faceNumber)
{ return (faceNumber == 12) ? 8 : faceNumber + 1; };
static face_t PrevSouthEqFace(face_t faceNumber)
{ return (faceNumber == 8) ? 12 : faceNumber - 1; };

// find next or previous (numerically) North Equatorial faces
static face_t NextNorthEqFace(face_t faceNumber)
{ return (faceNumber == 6) ? 2 : faceNumber + 1; };
static face_t PrevNorthEqFace(face_t faceNumber)
{ return (faceNumber == 2) ? 6 : faceNumber - 1; };

// tell which region a vertex is in
static bool IsNorthPoleVertex(vertex_t vertexNumber)
{ return (vertexNumber <= 4); };
static bool IsNorthEquatorVertex(vertex_t vertexNumber)
{ return (vertexNumber > 4 && vertexNumber <= 9); };
static bool IsSouthEquatorVertex(vertex_t vertexNumber)
{ return (vertexNumber > 9 && vertexNumber <= 14); };
static bool IsSouthPoleVertex(vertex_t vertexNumber)
{ return (vertexNumber > 14); };

// tell which region a face is in
static bool IsNorthPoleFace(face_t faceNumber)
{ return (faceNumber == 1); };
static bool IsNorthEquatorFace(face_t faceNumber)
{ return (faceNumber > 1 && faceNumber <= 6); };
static bool IsSouthEquatorFace(face_t faceNumber)
{ return (faceNumber > 6 && faceNumber <= 11); };
static bool IsSouthPoleFace(face_t faceNumber)
{ return (faceNumber == 12); };

// find non-adjacent South Equatorial faces holding
// these vertices. This is useful for lofting because
// we can rotate these two faces independently without
// affected the other face's vertex.
// The vertices v1 and v2 can be on the South Equator,
// the South Pole, or a mixture; except that they cannot
// be on the same vertical line (because they touch the
// same faces then); the caller must detect this and not
// call this routine in this case.
static void FindNonAdjacentSouthFaces(vertex_t v1,
vertex_t v2, face_t *pf1, face_t *pf2);

// write a comment line to the rotations file telling
// what we are doing (debug only)
void WriteComment(const char *pComment);

// check whether an edge has the given colors (in either order)
bool EdgeHasColors(face_t f0, face_t f1, color_t c0,
color_t c1);

private:

////////////////////////////////////////
// used in implementation

void SliceImp(face_t faceNumber, Direction direction);
// slice one turn

// utility for rotating part of a face
static void RotateFacelets(short **ppColor,
        Direction direction);

////////////////////////////////////////
// member variables

// colors for edge and corner facelets
// colors are numbered 1 through 6; we use 0 for an invalid entry.
//
// indices are the face number (and so run from 1 through 12).
// edge facelets are indexed by the two faces they touch, with

```



New OS. New Software. New Installer.

Introducing InstallAnywhere - Mac OS X Edition, the new power tool for your Mac OS X software installation. Stop using yesterday's tired old installer tools. Now, you can create industrial strength installers that are flexible, intuitive, and royalty-free. Your software will look better than ever and install perfectly, every time.

InstallAnywhere supports all Mac OS X features, like user authentication, file permissions, installing icons into the Dock, and offers a fully customizable Aqua look and feel.

Software innovators like Adobe, Apple, Borland, Gracion, LimeWire, and Sun already depend on Zero G for their software installation needs. See for yourself why InstallAnywhere - Mac OS X Edition is the new power tool for your software.

InstallAnywhere - The Industrial Strength Installer From Zero G.

Download a free trial version from <http://www.ZeroG.com/goto/mac>



```

// the first index being the face they are on.
// corner facelets are indexed by the three faces they touch,
// with the first index being the face they are on,
// with second index < third index.
//
// We allocate many more items than are actually valid.
short fEdgeFacelets[13][13];
short fCornerFacelets[13][13][13];

// output stream for the answer
std::ofstream fRotationsStream;

);

// class to temporarily rotate a face; when destructed,
// it rotates back in the other direction. The clicks
// can be 0, meaning no rotation.

class CTempRotate
{
public:
    CTempRotate(CMegaminx& rMega, CMegaminx::face_t faceNumber,
        int clicks, CMegaminx::Direction direction);

    ~CTempRotate();
private:
    CMegaminx& fMega;
    CMegaminx::face_t fFaceNumber;
    int fClicks;
    CMegaminx::Direction fDirection;
};

```

CSolver.h

```

//
// This class contains all the algorithms for solving Megaminx.
//
//
#pragma once

#include <fstream>
#include <string>
using std::string;

#include "CMegaminx.h"

class CCornerVisitor;
class CMegaminx;

class CSolver
{
public:
    CSolver(CMegaminx& rMega);
    ~CSolver();

    // solve the Megaminx and write out the solution
    void Solve();

    // call visitor
    void VisitAllCorners(CCornerVisitor &aVisitor);

private:
    // used in implementation

    // double operations from Meffert
    // RUU = R upper star upper star, and so on
    void DoLUU(CMegaminx::face_t leftFace,
        CMegaminx::face_t rightFace);
    void DoRUU(CMegaminx::face_t leftFace,
        CMegaminx::face_t rightFace);
    void DoRLl(CMegaminx::face_t leftFace,
        CMegaminx::face_t rightFace);
    void DoLLl(CMegaminx::face_t leftFace,
        CMegaminx::face_t rightFace);

    // distance along either the North or South pole vertices, measured
    // in the direction of increasing vertex number and wrapping around.
    // We use this when we are going to rotate in this direction.

```

```

// Also: distance along an equator from one face to another.
// This calculates (to - from) mod 5.
int Distance(int from, int to)
{ int dist = to - from; if (dist < 0) dist += 5; return
dist;};

```

```

// this checks that the South half edges are an even permutation
// of the solved position. It returns true if the permutation
// is even and false if it is odd.
bool CheckEdgeParity();

```

```

static int ParityLookup(CMegaminx::color_t c0,
    CMegaminx::color_t c1);

```

// solution steps

```

void Step3();
void Step3Edges();
CMegaminx::face_t Step3_4Drop(CMegaminx::color_t c0,
    CMegaminx::color_t c1);
void Step3Corners();
void Step4();
void Step5();
void Step5PlaceVertex(CMegaminx::vertex_t srcVertex,
    int destVertex);
void Step5OrientVertex(int destVertex);
void Step6();
void Step6PlacePoleEdge(CMegaminx::face_t fromSFace,
    CMegaminx::face_t toEdgeIndex);
void Step7();
void Step7PlacePoleEdge(CMegaminx::face_t srcFaceN,
    CMegaminx::face_t destFaceL,
    CMegaminx::face_t destFaceR);
void Step8();
void Step8ReferenceEdge();
void Step8SecondEdge();
void Step8ThirdEdge();
void Step8RestoreEquator();
void Step8OrientFourFive();
void Step9();
void Step9EquatorAndPole(CMegaminx::vertex_t dst,
    CMegaminx::vertex_t src);
void Step10();
void Step11();
void Step11BothEquators(CMegaminx::vertex_t vCounterCW,
    CMegaminx::vertex_t vCW);

```

```

// verification steps (these run only in debug mode);
// they check that various things are correctly positioned
// at the end of step n, and assert if they are not.
// Each step calls the preceding step, so all earlier
// verifications are re-performed too.

```

```

void Step3Verify();
void Step4Verify();
void Step5Verify();
void Step6Verify();
void Step7Verify();
void Step8Verify();
void Step9Verify();
void Step10Verify();
void Step11Verify();

```

////////////////////////////////////

// member variables

```
CMegaminx& fMega; // Megaminx being solved
```

```
};
```

// CCornerVisitor Class

```

class CCornerVisitor
{
public:
    CCornerVisitor() {};
    virtual ~CCornerVisitor() {};

    virtual void VisitCorner(int cornerIndex) = 0;
};

```



and print
The File Share Folks™



My files are
important.

Very important.

I only share my
files with those
I trust.

I trust DAVE.

Mac to PC, PC to Mac. Cross-platform file and print sharing is too vital to your business to risk. Trust Thursby, the company with 15 years experience. Trust DAVE, the solution with a proven track record. Share files and printers across a network with no barriers. DAVE installs on your Mac with no additional software required for the PC. It's fast, secure and easy to use. Download a free evaluation today!

Trust **DAVE®**.



**New! Support
for inkjet
printing**



www.thursby.com

Mac is a trademark of Apple Computer, Inc. registered in the U.S. and other countries. The "Built for Mac OS X" graphic is a trademark of Apple Computer, Inc., used under license.
DAVE is a registered trademark of Thursby Software Systems, Inc.
© 2002, Thursby Software Systems, Inc.

By Andrew S. Downs

Dock Tile Imaging

Changing a Java application's dock tile at runtime

OVERVIEW

The Dock Manager API allows a programmer to alter the tile for an application at runtime. Using the Java Native Interface, a Java application can change its tile dynamically as well. This involves a combination of Java and native code.

Two approaches are illustrated in this article. The first captures the pixels from a Java image and passes them to a native library function, which uses the CoreGraphics (Quartz) API to replace the application's tile (see **Figure 1**).



Figure 1. A modified Dock tile.

The second example uses QuickDraw to paint a progress bar over the tile, as shown in **Figure 2**. The progress data is sent from Java to a native library function, and the imaging is done in the library.



Figure 2. The progress bar at the bottom of the tile, showing 60% complete.

If you want to code along with the examples, my recommendation is to first download the JNISample project from Apple's developer site (see the URLs at the end of this article). That project served as the structural basis for the code in this article. All the build settings are already in place, making it an easy-to-use learning tool. (There are targets for compiling both the Java and native code, generating the javah file, building a library, etc.) I kept the filenames the same, but replaced the content of the various files. In the listings below you will see the filenames as they exist in that project.

JAVA

One class (DockTiler) provides most of the functionality for this example. It relies on two other classes for getting and drawing (offscreen) an image from a local file. You can also load an image via a non-local URL, which would allow the app to change the tile in response to outside conditions. For example, an application that retrieves weather data can change the tile to reflect current conditions or the forecast.

The JNISample class contains main(), the entry point for the Java application. It is used simply to instantiate DockTiler and invoke one of its instance methods.

Listing 1: JNISample.java

JNISample

The classes contained here include:

JNISample: creates and calls a DockTiler instance.

DockTiler: loads the image and sends it to the native drawing code.

LocalFile: allows the user to select a local file for display in the tile.

PictureFrame: an offscreen Canvas which draws the image, allowing DockTiler to retrieve the image pixels.

//The image support comes from the AWT and the image package.

```
import java.awt.*;
import java.awt.image.*;
import java.util.*;
```

```
public class JNISample {
    public JNISample() {}
```

//Test code.

```
public static void main (String args[]) {
    DockTiler dock = new DockTiler();
```

```
    dock.test();
```

```
    System.out.println( "Finished." );
```

```
}
```

Andrew has been a Java fan since his first encounter with the language at the WebEdge III conference in 1996. You can reach him at andrew@downs.ws.

Without this piece, you're not done.



It's not enough just to write solid code anymore. Users expect an easy-to-use installer when they buy your software.

InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.

Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers. Even create demoware in minutes.

Visit <http://www.stuffit.com/installermaker/> to also learn about creating Mac® OS X compatible installers with InstallerMaker.

Then you're done.

Stuffit InstallerMaker

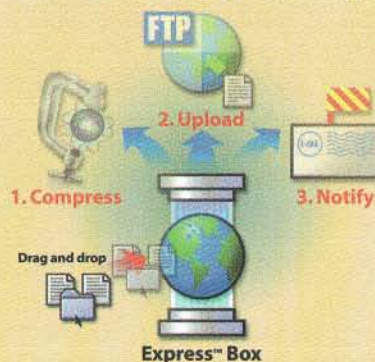
The complete installation and distribution solution.™

Get your beta testing started faster.

Guarantee your builds arrive safe and sound! New **Stuffit Express™** automates file compression and transfer tasks to eliminate costly errors every time you exchange data. By creating custom drop box applications, you and those you communicate with can accomplish complex file transfer tasks with a simple drag and drop. Choose from up to 26 file transfer steps with no coding required. Boost

productivity and save time by distributing your custom Express boxes to everyone on your list.

<http://www.stuffit.com/express/enterprisemac>.



```

class DockTiler {
    // If we have trouble loading an image, the values of its width and height will
    // remain at -1. See loadImage().
    int mWidth = -1, mHeight = -1;
    int mPixels[];

    static {
        // Load the library when this class gets loaded.
        System.loadLibrary( "Example" );
    }

    public DockTiler() {

        // These are the two functions in the shared library that we will call.
        // Note the declaration as native.
        native void setDockTile( int[] pixels, int width,
            int height );

        native void updateProgressBar( int currPercent );

        // The primary test driver.
        public void test() {
            loadImage();

            setDockTile( mPixels, mWidth, mHeight );

            performTask();

        }

        // Read in the image and retrieve its pixels.
        protected void loadImage() {
            LocalFile lf = new LocalFile();
            String filename = lf.getFilePath();

            Image image =
                Toolkit.getDefaultToolkit().getImage( filename );

            // Send the image to the Canvas, where it will be rendered.
            PictureFrame pf = new PictureFrame( image );

            Frame f = new Frame( "Image" );

            // Setup offscreen.
            f.setBounds( -250, -250, 200, 200 );

            f.setLayout( new BorderLayout() );
            f.add( "Center", pf );
            pf.setSize( 128, 128 );
            f.pack();

            // An invisible window won't render an image.
            f.setVisible( true );

            f.repaint();
            pf.repaint();

            // Use the Canvas as the observer during the loading process.
            int width = image.getWidth( pf );
            int height = image.getHeight( pf );

            // Allocate storage for the image data.
            mPixels = new int[ width * height ];

            // Create an object to copy the image pixel data into our array.
            PixelGrabber pg = new PixelGrabber( image, 0, 0,
                width, height, mPixels, 0, width );

            // Copy the pixels to the array.
            try {
                pg.grabPixels();

                // Check for error using bit values in the ImageObserver class.
                if ( ( pg.getStatus() & ImageObserver.ABORT ) != 0 )

                    return;

                // If successful, set instance attributes to legitimate values (not -1).
                mWidth = width;
                mHeight = height;
            }
        }
    }

```

```

    }
    catch ( InterruptedException e ) {
        return;
    }
}

// For a task that may take some time, it helps to wrap it in a separate method
or
// even class, and spin it off as a thread. Here, simply get the current progress
// and display it.
protected void performTask() {
    int percentComplete = 0;

    boolean taskComplete = false;

    while ( !taskComplete ) {
        // Call the native method that draws the progress bar.
        updateProgressBar( percentComplete );

        percentComplete = updateTask();

        if ( percentComplete >= 100 )
            taskComplete = true;
    }

    int mCount = 0;

    // Lengthy tasks will use a sophisticated approach to determining completion.
    // This example uses a simple loop so we can watch the bar move.
    protected int updateTask() {
        return mCount++;
    }
}

class LocalFile {
    FileDialog mFileDialog = null;

    // Display a dialog asking the user to choose a file.
    public LocalFile() {
        if ( mFileDialog == null ) {
            mFileDialog = new FileDialog( new Frame(),
                "Select an image file", FileDialog.LOAD );
        }
    }

    // Build and return the path to the file (if selected).
    public String getFilePath() {
        mFileDialog.setVisible( true );

        String retval = "";

        if ( mFileDialog.getFile() != null &&
            mFileDialog.getFile().length() > 0 ) {
            retval = mFileDialog.getDirectory();

            if ( !retval.endsWith(
                System.getProperty( "file.separator" ) ) )
                retval += System.getProperty( "file.separator" );

            retval += mFileDialog.getFile();
        }

        return retval;
    }
}

// A subclass of java.awt.Canvas that draws an Image object.
class PictureFrame extends Canvas {
    Image mImage;

    PictureFrame( Image img ) {
        super();
        mImage = img;
        setBackground( Color.white );
    }

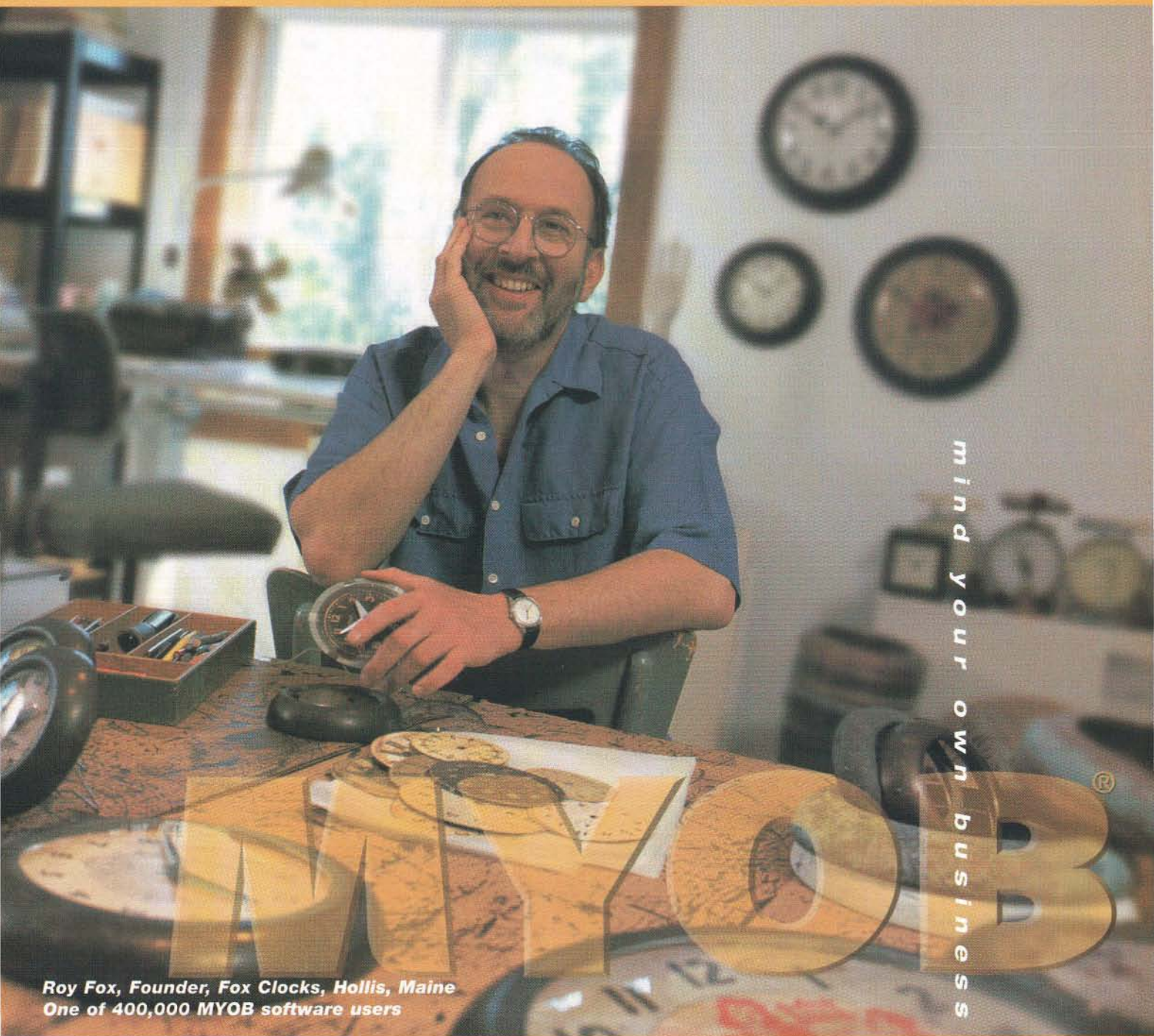
    public void update( Graphics g ) {
        paint( g );
    }
}

```

DESIGNED FOR THE WAY YOU WORK

I'd rather create clocks than invoices.

If I wanted to keep books all day, I'd have been an accountant.



Roy Fox, Founder, Fox Clocks, Hollis, Maine
One of 400,000 MYOB software users

MYOB

**Small Business.
Smart Solutions.SM**

AccountEdge on the Mac.
Software includes QuickBooks[®] conversion
utility. MYOB also offers conversion service.

800.322.MYOB
myob.com/us

®QuickBooks is a trademark of Intuit, Inc.

MYOB software is the simplest, most powerful, most complete solution for managing my company on the Mac, from the day to day to the bottom line.

Antique frames. Quartz movements. That's my business.

MYOB software works for me.

mind your own business[®]

```
// Draw the image.
public void paint( Graphics g ) {
    g.drawImage( mImage, 0, 0, this );
}
}
```

JAVA NATIVE INTERFACE CODE

JNI code is C code that bridges the Java and native worlds, handling the conversion between Java data types and their native counterparts. The JNIEnv pointer in each function indirectly points to a function table containing JNI functions, and the jobject references the instance of the class making the call. Any additional arguments are the values passed from the Java code to the native code.

Listing 2: ExampleJNILib.c

ExampleJNILib

The JNI glue for converting arguments prior to calling through to the native code.

```
#include "JNISample.h"
#include "ExampleDylib.h"

JNIEXPORT void JNICALL Java_DockTiler_setDockTile(
    JNIEnv *env, jobject this, jintArray pixels, jint
    width, jint height ) {
    // Obtain a pointer to the array to pass to the native function.
    jint *theArray = (*env)->GetIntArrayElements(
        env, pixels, NULL );

    if ( theArray != NULL ) {
        // Call the library function.
        // Note that no adjustments are made to the primitive values.
        setDockTile( theArray, width, height );

        // Tell the VM we are no longer interested in the array.
        (*env)->ReleaseIntArrayElements( env, pixels,
            theArray, 0 );
    }
}

JNIEXPORT void JNICALL Java_DockTiler_updateProgressBar(
    JNIEnv *env, jobject this, jint currPercent ) {
    // Call the library function.
    // No additional translation is needed on primitive values.
    updateProgressBar( currPercent );
}
```

THE LIBRARY

The native library contains the actual tile drawing code. One function creates an image and replaces the existing tile with the new one. The second function uses a completion percentage value to determine how much of the progress bar to paint. It draws over the bottom of the existing tile.

Listing 3: ExampleDylib.c

ExampleDylib.c

Perform drawing in the Dock tile.

```
#include <stdio.h>
#include <Carbon/Carbon.h>
#include <ApplicationServices/ApplicationServices.h>
```

// Args include the array of pixel RGBA values, and the actual image width and height.

```
extern void setDockTile( int * imagePixels, int width,
    int height ) {
    // How many bytes in each pixel? Java uses 4-byte ints.
    int kNumComponents = 4;
```

```
    OSStatus theError;
```

```
    // Several CoreGraphics variables.
    CGContextRef theContext;
    CGDataProviderRef theProvider;
    CGColorSpaceRef theColorspace;
    CGImageRef theImage;
```

```
    // How many bytes in each row?
    size_t bytesPerRow = width * kNumComponents;
```

```
    // Obtain graphics context in which to render.
    theContext = BeginCGContextForApplicationDockTile();
```

```
    if ( theContext != NULL ) {
        // Use the pixels passed in as the image source.
        theProvider = CGDataProviderCreateWithData(
            NULL, imagePixels, ( bytesPerRow * height ), NULL
        );
```

```
        theColorspace = CGColorSpaceCreateDeviceRGB();
```

```
        // Create the image. This is similar to creating a PixMap.
        // - The width and height were passed as arguments.
        // - The next two values (8 and 32) are the bits per pixel component and
        //   total bits per pixel, respectively.
        // - bytesPerRow was calculated above.
        // - Use the colorspace ref obtained previously.
        // - The alpha or transparency data is in the first byte of each pixel.
        // - Use the data source created a few lines above.
        // - The remaining parameters are typical defaults. Consult the API docs for
        //   more info.
```

```
        theImage = CGImageCreate( width, height, 8, 32,
            bytesPerRow, theColorspace, kCGImageAlphaFirst,
            theProvider, NULL, 0, kCGRenderingIntentDefault );
```

```
        CGDataProviderRelease( theProvider );
        CGColorSpaceRelease( theColorspace );
```

```
        // Set the created image as the tile.
        theError = SetApplicationDockTileImage( theImage );
```

```
        CGContextFlush( theContext );
```

```
        CGImageRelease( theImage );
```

```
        EndCGContextForApplicationDockTile( theContext );
    }
```

```
extern void updateProgressBar( const int currPercent ) {
    CgrafPtr thePort;
    Rect theRect;
    float right = 0;
```

```
    // Obtain graphics context.
    thePort = BeginQDContextForApplicationDockTile();
```

```
    if ( thePort != NULL ) {
        // Good ol' QuickDraw.
        GetPortBounds( thePort, &theRect );
```

```
        // Initially, draw the background of the bar and frame it.
```

```
        if ( currPercent == 0 ) {
            SetRect( &theRect, theRect.left, theRect.bottom -
                10,
                theRect.right, theRect.bottom );
            ForeColor( redColor );
            PaintRect( &theRect );
            ForeColor( blackColor );
            FrameRect( &theRect );
        }
    }
```

```
// Calculate right-edge of progress bar.
if ( currPercent >= 100 )
    right = ( float )theRect.right;
else
    right = ( ( ( float )theRect.right -
        ( float )theRect.left ) /
        ( float )100 ) * ( float )currPercent;

// Draw the entire progress bar up until this point.
ForeColor( greenColor );

// Inset the progress rectangle on our own.
SetRect( &theRect, theRect.left + 1,
    theRect.bottom - 9, ( int )right,
    theRect.bottom - 1 );

PaintRect( &theRect );

QDFlushPortBuffer( thePort, NULL );

EndQDContextForApplicationDockTile( thePort );
}
```

The image creation using the CoreGraphics API is the trickiest part. This example uses the fact that a Java int is 4 bytes, and that alpha (transparency) data, if included, is stored in the most significant byte. After some trial and error, I found that the settings shown here work for the images I tested against.

USEFUL URLS

I used quite a few outside sources (primarily Apple) in preparing this article. Though some of these URLs may change, I want to at least point you in the right direction.

- The tile images came royalty-free from The Clip Art Connection:
<http://www.clipartconnection.com/photos/index.html?gid=18937>
- Sun has a simple PixelGrabber example:
<http://java.sun.com/products/java-media/2D/forDevelopers/2Dapi/java/awt/image/PixelGrabber.html>
- The QuickDraw API:
<http://developer.apple.com/techpubs/macosx/Carbon/graphics/QuickDraw/quickdraw.html>
- The Dock Manager API:
<http://developer.apple.com/techpubs/macosx/Carbon/HumanInterfaceToolbox/DockManager/dockmanager.html>
- The CoreGraphics routines are briefly discussed in the QuartzPrimer:
<http://developer.apple.com/techpubs/macosx/Essentials/QuartzPrimer.pdf>
- A sample JNI application that formed the basis for the project in this article:
http://developer.apple.com/samplecode/Sample_Code/Java/JNISample.htm
- A sample Dock drawing application (in C):
http://developer.apple.com/samplecode/Sample_Code/Human_Interface_Toolbox/Tiler.htm
- This one is a book, not a URL, and is very useful:
Liang, Sheng. The Java Native Interface. Sun Microsystems, Inc. 1999.

How IMPORTANT IS Your DATA?

With its redundant, multi-file design,

OpenBase SQL

ensures data integrity and availability,

while enhancing database performance.



OPENBASE®
INTERNATIONAL

For more info, visit:

www.openbase.com

by Tim Monroe

Virtuosity

Programming with QuickTime VR

INTRODUCTION

QuickTime VR (or, more briefly, *QTVR*) is the part of QuickTime that allows users to interactively explore and examine photorealistic, three-dimensional virtual worlds and objects. A QuickTime VR movie is a collection of one or more *nodes*; each node is either a *panoramic node* (also known as a *panorama*) or an *object node* (also known as an *object*). **Figure 1** shows a view of a sample panoramic node, and **Figure 2** shows a view of an object node. (When a QuickTime VR movie consists of a single node, folks often refer to it as a *panorama movie* or an *object movie*, depending on the type of node it contains.)

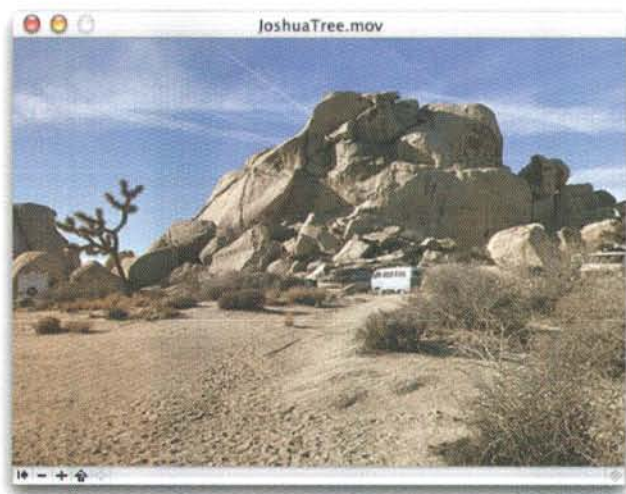


Figure 1: A QuickTime VR panorama



Figure 2: A QuickTime VR object movie

QuickTime VR movies are managed by the *QuickTime VR movie controller*, a movie controller component that knows how to interpret user actions in a QuickTime VR movie. The QuickTime VR movie controller also displays a controller bar with buttons that are appropriate to QuickTime VR movies. From left to right, the five buttons allow the user to go back to the previous node, zoom out, zoom in, show the visible hot spots, and translate an object in the movie window. The QuickTime VR movie controller automatically disables any buttons that are not appropriate for the current node type or movie state. For instance, the back button is disabled in Figure 2 because the movie is a single-node movie. Similarly, the translate button is disabled in Figure 1 because the current node is a panoramic node, not an object node.

QuickTime has supported QuickTime VR movie creation and playback since mid-1995. In early 1997, Apple released QuickTime VR version 2.0, which (in addition to numerous other improvements) provided a C programming interface to QuickTime VR. This interface, called the *QuickTime VR Manager*, provides an extensive set of functions for controlling QuickTime VR movies. In this article, we'll take a look at the QuickTime VR Manager.

The QuickTime VR movie controller also allows QuickTime VR movies to send and receive wired actions. This allows us, for instance, to use buttons in a Flash track to control a QTVR movie, as illustrated in **Figure 3**. Here the Flash buttons in the lower-left corner of the movie are configured to send the appropriate QuickTime wired actions to pan,

Tim Monroe is a member of the QuickTime engineering team. You can contact him at monroe@apple.com. The views expressed here are not necessarily shared by his employer.



REAL Software and MacTech present the REALbasic Showcase to highlight some of the fantastic solutions created by REALbasic users worldwide. The showcase illustrates the wide range of applications that developers using REALbasic can create. Some benefit any Mac user, and others are more specific. All of them are seriously cool!

REALbasic is the powerful, easy-to-use tool for creating your own software for Macintosh, Mac OS X, and Windows. It runs natively on Mac OS X as well as earlier versions of the Mac OS. For more information, please visit: [<www.realbasic.com>](http://www.realbasic.com).

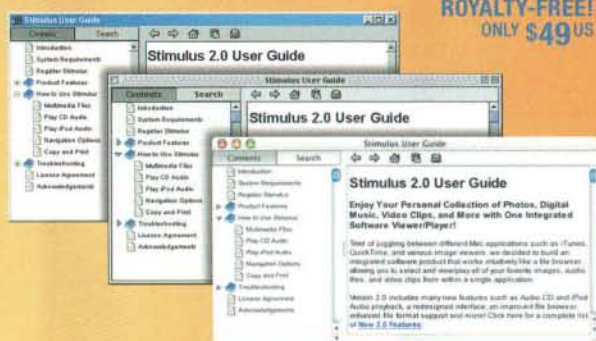
The Made with REALbasic program is a cooperative effort between REALbasic users and REAL Software, Inc. to promote the products created using REALbasic and the people who create them. For more information about the Made with REALbasic program, please visit: [<www.realbasic.com/realbasic/mwrb/Partners/MwRbProgram.html>](http://www.realbasic.com/realbasic/mwrb/Partners/MwRbProgram.html).

UniHelp

Cross-Platform HELP
Module for REALbasic

Apple Help Viewer, Microsoft HTML Help, WinHelp.... Why waste your precious development time designing and compiling a different Help system for each platform when you can easily use UniHelp for all of them?

NO additional plugins or classes needed. Just drag the UniHelp module and its graphics folder into your REALbasic project, add a few lines of code, create your external help pages, and that's it – instant online help for your compiled REALbasic applications that looks and functions the same on **Classic Mac, Mac OS X and Windows 98/NT/ME/2000/XP**.



ROYALTY-FREE!
ONLY \$49 US

KEY FEATURES:

- Displays Both HTML and ASCII Text Files.
- Supports URL Links, Relative Links, Anchors & Email Links.
- Parses More Than 20 HTML Tags, including .
- Supports Images, Video and Audio.
- Built-in Search Engine.
- Hierarchical Listbox Displays Your Help Table of Contents.
- Customizable Listbox Icons, Window Title, Start Page, Font, Contents Sequencing, etc.
- GUI Support for 6 Languages: English, Spanish, French, Italian, Portuguese and German.
- Small Footprint: Since Your Help Pages are External, only the UniHelp Module is Compiled with Your REALbasic Applications.
- Familiar Help-style Interface with Back, Forward, Home, Copy and Print buttons.
- Easy to Use & Royalty-Free!

"UniHelp" and "Electric Butterfly" are never mentioned anywhere on the UniHelp interface, providing you with a full-featured generic Help solution – your customers will think you built it yourself!



electric
butterfly

FREE DOWNLOAD!
<http://www.ebutterfly.com>

db Reports

Printing database driven reports from your REALbasic project has never been easier.

Powerful expressions with an easy-to-use interface make this an indispensable tool for the REALbasic programmer providing business solutions.

Download a free demo today.

<http://abDataTools.com>

ftp> get pxFTP

The power and stability of the command line...
The ease of drag and drop!



Get it today at...
PIDOG.COM

From the maker of...

piDock

TelnetLauncher

π

piDog Software



Picture Play

Create digital image compositions on your Mac, quickly and easily.



For more information,
email chris@crescendosw.com, or
visit us today at www.crescendosw.com!

Extend REALbasic with Advanced Plugins

Spreadsheet Controls:

We provide a wide range of spreadsheet controls for REALbasic, including multistyled and custom rendering spreadsheet controls.

A	B	C
This is some incredible list		
1	Some text	More text
2	Some text	More text
3	Some text	More text
4	Some text	More text
5	Some text	More text
6	Some text	More text
7	Some text	More text
8	Some text	More text

- More than 32k of rows.
- Classic, OS X and Win32.
- Accelerated for maximum speed.
- Images in cells.



0110010

Cryptography:

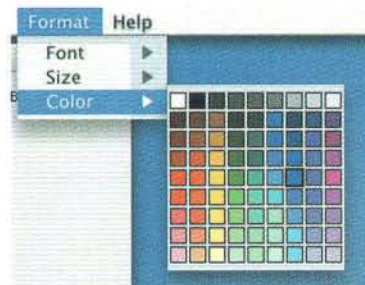
We provide data encryption, encoding, compression and hashing for REALbasic.

Supported Algorithms:

- Encryption:
 - e-CryptIt
 - BlowFish (448 bit)
 - AES (Rijndael) (256 bit)
- Encoding:
 - e-CryptIt Flexible
 - Base 64
 - BinHex
 - MacBinary III
 - AppleSingle / Double
 - UUcoding
- Compression:
 - Zip on Strings and streams (.gz)
- Hashing and Checksums:
 - CRC32 / Adler32
 - MD5 / HMAC_MD5
 - SHA / SHA1 / HMAC_SHA1

Other Plugins:

We have many other plugins for REALbasic, including plugins to do advanced MacOS Toolbox tasks and more custom Controls.



Speed up development and make more advanced applications by using plugins ! Get free demos at www.einhugur.com



Einhugur Software
sales@einhugur.com
www.einhugur.com

Corona

accounting software



"Easy to set up, easy to use, and excellent support from the developer."

Five stars on VersionTracker.com
Four cows on Tucows.com

- cash entry
- invoicing
- payroll
- reports

\$64.95

Free 30-day trial

<http://homepage.mac.com/idlewild/CoronaUS.hqx>

Version 1.9 now with: sales tax accounting
"keyless entry" invoices
drag 'n drop accounts



A best friend for business!

p.o. box 472 • aurora • oregon • 97002
idlewild@mac.com

iScreensaver Designer

for Macintosh and Windows



Mac designers...
build professional screensavers
without touching a Windows machine

the world's only cross-platform screensaver design tool

- build both Macintosh and Windows screensavers with a single click, on whatever system* you are using!
- use any QuickTime movie format :
Macromedia Flash, MPEG, Cinepak, MP3, Midi, AVI, DV Video...
- include a hidden movie that can be unlocked with a registration code
- customize and fully-brand both Installers and Screensaver control panels with pictures and text
- screensavers install without DLLs, extensions, or restarts

simple
WYSIWYG
editor

supports
interactive Flash
and QuickTime

consistent
cross-platform
user interface

try before you buy
fully functional
online downloads



the iScreensaver Designer editing environment

<http://iScreensaver.net>

email : info@iScreensaver.net

* supported systems, as of November 2001, include:
Macintosh OS 8.6 to 9.2, Microsoft Windows 95/98/ME, NT4/NT2000/XP

©2001 Xochi Media Inc. All Rights Reserved.
iScreensaver Designer and the DoubleClick logo are trademarks of Xochi Media Inc.
Apple, Macintosh, and QuickTime are trademarks and/or registered trademarks of Apple Computer, Inc.
Macromedia Flash is a trademark of Macromedia, Inc. Microsoft and Windows are registered trademarks of Microsoft Corporation.
All other trademarks are the property of their respective owners. Made using REALbasic.

Whistle Blower

Enterprise server monitor and restart utility
whistleblower.sentman.com

Connect to and validate the response from web servers,
cgi scripts and over 23 other types of servers.

Send email, pages and perform unattended restarts via
MasterSwitch or PowerKey.

Shifts make sure that the person on call when the server
goes down is the one who gets the page.

68k, PPC and Carbon

Web based administration lets you check on and restart
your servers from anywhere.

Customize your response to an outage with Apple Script.

email us at whistleblower@sentman.com

TitleTrack™ JUKEBOX

MAC MUSIC MANAGEMENT FOR
SONY® 5 TO 400 DISC CD CHANGERS



Control up to 12 changers (4,800 music CDs)
Control Any Brand Stereo Receiver
Play MP3 and other Sound Files
Plus much, much more!!!

www.titletrack.com

info@titletrack.com

tilt, or zoom the panorama. (We saw how to attach wired actions to Flash track buttons in “The Flash II: Revenge of the Trickster”, *MacTech*, February 2002.)

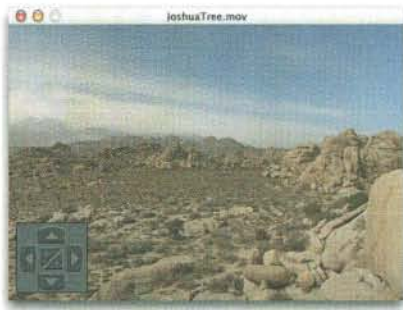


Figure 3: A Flash track controlling a QuickTime VR movie

We'll take a look at the actions that can be targeted at a QuickTime VR movie, and we'll also see how to attach wired actions to elements in a QuickTime VR movie. We can attach wired actions to a particular node or to a particular hot spot in a node. So, for example, we could wire a hot spot to launch the user's web browser and navigate to a particular web site when the user clicks that hot spot. Or we can have some actions triggered when the user enters a node. We won't learn how to build QuickTime VR movies in this article, but we will need to understand some of the structure of these movies in order to learn how to attach wired actions to nodes and hot spots.

THE QUICKTIME VR MANAGER

The QuickTime VR Manager provides a large number of capabilities that we can use to customize and extend the user's virtual experience of panoramas and objects. Here we'll summarize the basic capabilities of the QuickTime VR Manager. Then, in the following sections, we'll illustrate how to use some of them. The QuickTime VR Manager provides these main capabilities:

- **Positioning.** A QuickTime VR movie file contains a *scene*, which is a collection of one or more nodes. Each node is uniquely identified by its node ID. Within a panoramic node, the user's view is determined by three factors: the pan angle, the tilt angle, and the vertical field of view (sometimes also called the *zoom angle*). For objects, the view is also determined by the view center (the position of the center of the object in the movie window). The QuickTime VR Manager provides functions to get and set any of these items. For instance, we can programmatically spin an object around by repeatedly incrementing the current pan angle.
- **Hot spot handling.** We can use the QuickTime VR Manager to manage any hot spots in a panorama or object. For instance, we can trigger a hot spot programmatically (that is, simulate a click on the hot spot), enable and disable hot spots, determine whether the cursor is over a hot spot, find all visible hot spots, and so forth. We can also install a callback routine that is called whenever the cursor is over an enabled hot spot.
- **Custom node-entering and -leaving behaviors.** The QuickTime VR Manager allows us to perform actions whenever the user enters a new node or leaves the current node. For instance, we might use a node-entering procedure to play a

sound when the user enters a particular node. Or, we can use a node-leaving procedure to prevent the user from leaving a node until some task has been accomplished.

- **Getting information.** We can use the QuickTime VR Manager to get information about a scene or about a specific node. For instance, we might want to determine the ID and type of the current node. Much of the information about scenes and nodes is stored in atoms in the movie file. To get information about a scene or node that isn't provided directly by the QuickTime VR Manager, we'll need to use the QuickTime atom container functions to extract information from those atoms.
- **Intercepting QuickTime VR Manager functions.** We can intercept calls to some QuickTime VR Manager functions in order to augment or modify their behavior. For example, to assign behaviors to custom hot spots, we can install an intercept routine that is called whenever a hot spot is triggered. Our intercept routine might check the type of the triggered hot spot and then perform the actions appropriate for that type. Another common use of intercept routines is to intercept positioning functions (changing the pan, tilt, and field of view) and adjust environmental factors accordingly. For instance, we can adjust the balance and volume of a sound as the pan angle changes in a panorama, thereby making it appear that the sound is localized within the panorama.
- **Accessing the prescreen buffer.** QuickTime VR maintains an offscreen buffer for each panorama, called the *prescreen buffer*. The prescreen buffer contains the image that is about to be copied to the screen. We can use QuickTime VR Manager functions to access the prescreen buffer, perhaps to draw a graphic image over the panorama.

This list is not exhaustive. The QuickTime VR Manager provides many other capabilities as well. For a complete description, see the technical documentation cited at the end of this article.

QUICKTIME VR MOVIE PLAYBACK

Our existing sample applications, such as QTShell, are already able to open and display QuickTime VR movies. The QuickTime VR movie controller handles the basic click-and-drag navigation, keyboard input, and controller bar events. We need to use the QuickTime VR Manager only if we want to exploit some of the capabilities described just above.

Initializing the QuickTime VR Manager

Before we can call the QuickTime VR Manager, however, we need to do a little setting up (over and above what's required for using QuickTime). First, we need to ensure that the QuickTime VR Manager is available in the current operating environment. There are several Gestalt selectors that we can use to see whether the QuickTime VR Manager is available and what features it has. **Listing 1** shows the definition of the `QTVRUtils_IsQTVRMgrInstalled` function, which indicates whether the QuickTime VR Manager is available in the current operating environment.

Listing 1: Determining whether the QuickTime VR Manager is available

```

QTVRUtils_IsQTVRMgrInstalled
Boolean QTVRUtils_IsQTVRMgrInstalled (void)
{
    Boolean    myQTVRAvail = false;

```

```

long      myAttrs;
OSErr     myErr = noErr;
myErr = Gestalt(gestaltQTVRMgrAttr, &myAttrs);
if (myErr == noErr)
    if (myAttrs & (1L << gestaltQTVRMgrPresent))
        myQTVRAvail = true;
return(myQTVRAvail);
}

```

For simplicity, we'll introduce a global variable to keep track of whether the QuickTime VR Manager is available:

```
gQTVRMgrIsPresent = QTVRUtils_IsQTVRMgrInstalled();
```

On Windows operating systems, we need to call the InitializeQTVR function to initialize the QuickTime VR Manager, like this:

```

#ifdef TARGET_OS_WIN32
    InitializeQTVR();
#endif

```

We also need to close our connection to the QuickTime VR Manager before our application terminates:

```

#ifdef TARGET_OS_WIN32
    TerminateQTVR();
#endif

```

Calling any other QuickTime VR Manager functions before calling InitializeQTVR will result in an error on Windows.

Getting the QTVR Instance

The QuickTime VR Manager keeps track of QuickTime VR movies using an identifier called a *QTVR instance* (of data type QTVRInstance). Virtually all QuickTime VR Manager functions operate on QTVR instances. You can think of an instance as representing a scene — that is, a collection of nodes — or sometimes just the current node. We obtain a QTVR instance by calling the QTVRGetQTVRInstance function. QTVRGetQTVRInstance takes a reference to a QTVR track, which we can obtain by calling QTVRGetQTVRTrack. **Listing 2** shows our definition of QTApp_SetupWindowObject, which we call for every movie we open.

Listing 2: Getting a QTVR instance

```

void QTApp_SetupWindowObject (WindowObject theWindowObject)
{
    Track          myQTVRTrack = NULL;
    Movie          myMovie = NULL;
    MovieController myMC = NULL;
    QTVRInstance   myInstance = NULL;
    if (theWindowObject == NULL)
        return;
    // make sure we can safely call the QTVR API
    if (!gQTVRMgrIsPresent)
        return;
    // find the QTVR track, if there is one
    myMC = (**theWindowObject).fController;
    myMovie = (**theWindowObject).fMovie;
    myQTVRTrack = QTVRGetQTVRTrack(myMovie, 1);
    QTVRGetQTVRInstance(&myInstance, myQTVRTrack, myMC);
    (**theWindowObject).fInstance = myInstance;
    // do any QTVR window configuration
    if (myInstance != NULL) {
        // set unit to radians
        QTVRSetAngularUnits(myInstance, kQTVRRadians);
    }
}

```

Notice that we keep track of the QTVR instance by storing it in the fInstance field of the window object associated with the movie (here, theWindowObject). This gives us an easy way to determine whether a

given movie window contains a QuickTime VR movie. Notice also that we call the QTVRSetAngularUnits function to set our preferred angular units to radians. The QuickTime VR Manager can work with either degrees or radians when specifying angular measurements (for instance, when we call QTVRGetPanAngle). The default angular unit type is degrees. Internally, the QuickTime VR Manager always uses radians, and in some situations it gives us measurements in radians no matter what the current angular unit. In general, therefore, I find it easier to work in radians most of the time, so I've reset the angular unit type to radians. (Your preference may vary.) We can define some simple macros to allow us to convert between degrees and radians:

```

#define kVRPi ((float)3.1415926535898)
#define kVR2Pi ((float)(2.0 * 3.1415926535898))
#define QTVRUtils_DegreesToRadians(x) \
    ((float)((x) * kVRPi / 180.0))
#define QTVRUtils_RadiansToDegrees(x) \
    ((float)((x) * 180.0 / kVRPi))

```

We don't need to explicitly release or dispose of a QTVR instance; the value we obtain by calling QTVRGetQTVRInstance remains valid until we dispose of the associated movie controller.

Controlling View Angles

Finally we're ready to use the QuickTime VR Manager to do some real work. The most basic way to use the API is to control the view angles of a node — the pan, tilt, and field of view angles.

Listing 3 defines a function that gradually increments the pan angle through 360 degrees. With panoramas, this has the effect of making the user seem to spin a full circle (as if the user were spinning on a rotating stool). With objects, this has the effect of making the object spin around a full circle (as if the object were spinning on a turntable).

Listing 3: Spinning a node around once

```

void SpinAroundOnce (QTVRInstance theInstance)
{
    float myOrigPanAngle, myCurrPanAngle;
    myOrigPanAngle = QTVRGetPanAngle(theInstance);
    for (myCurrPanAngle = myOrigPanAngle;
         myCurrPanAngle <= myOrigPanAngle + kVR2Pi;
         myCurrPanAngle += QTVRUtils_DegreesToRadians(10.0)) {
        QTVRSetPanAngle(theInstance, myCurrPanAngle);
        QTVRUpdate(theInstance, kQTVRCurrentMode);
    }
}

```

The idea here is simple: get the starting pan angle (by calling QTVRGetPanAngle) and then repeatedly increment the pan angle by a certain amount (here, 10 degrees) until a full circle has been traversed. Note that we need to call the QTVRUpdate function after we set a new pan angle to make sure the updated view is displayed on the screen.

Drawing on a Panorama

Suppose we want to draw a logo or other graphic element on top of a panorama (as seems to be in vogue on broadcast television channels these days). As we learned earlier, we can draw into a panorama's prescreen buffer before that buffer is copied to the screen. (Object nodes don't have prescreen buffers, so this technique won't work for those kinds of nodes.) We exploit this capability by installing a *prescreen buffer imaging completion procedure*, which is called by the QuickTime VR Manager each time the prescreen buffer

Holds up to 256MB and fits on your key chain or in the smallest pocket so you can transport your files anywhere!

The first nonvolatile FireWire™ flash key chain storage solution. Supports CompactFlash™ and IBM MicroDrives™.

KeyChain™



USB Flash Key™



portable storage

FireFly™

SMARTDISK
Simplifying The Digital Lifestyle™



Holds up to 5GB of anything you choose! The FireWire™ connection allows for a transfer rate of 12MB per second.

wiebeTECH



MicroGB™

Has the capacity to hold up to 48GB of information and offers both FireWire™ and USB interface.

Dr. Bott

www.drbott.com 877.611.2688

RESELLERS! Hundreds of Mac-friendly products available through Dr. Bott

is about to be copied to the screen. We install our procedure using the `QTVRSetPrescreenImagingCompleteProc` function:

```
ImagingCompleteUPP myImagingProc;
myImagingProc = NewImagingCompleteProc(MyPrescreenRoutine);
QTVRSetPrescreenImagingCompleteProc(myInstance,
    myImagingProc, (SInt32)theWindowObject, 0);
```

The `QTVRSetPrescreenImagingCompleteProc` function takes four parameters, which are the QTVR instance, a universal procedure pointer to the imaging complete procedure, a four-byte reference constant, and four-byte flags parameter. In this case, we pass the window object reference as the third parameter so that the imaging complete procedure can access any data associated with the window.

Our prescreen buffer imaging completion procedure is called after QuickTime VR has finished drawing into the prescreen buffer. When it's called, the current graphics port is set to the prescreen buffer. All we need to do is draw a picture at the appropriate spot, as shown in **Listing 4**.

Listing 4: Drawing a picture on top of a panorama

```
MyPrescreenRoutine
pascal OSErr MyPrescreenRoutine
    (QTVRInstance theInstance, WindowObject theWindowObject)
{
#pragma unused(theInstance)
    ApplicationDataHdl myAppData;
    Rect myMovieRect;
    Rect myPictRect;
    // get the application-specific data associated with the window
    myAppData = (ApplicationDataHdl)
        GetAppDataFromWindowObject(theWindowObject);
    if (myAppData == NULL)
        return(paramErr);
    // if there is no picture to display, just return
    if ((*myAppData).fPicture == NULL)
        return(noErr);
    // get the current size of the movie
    GetMovieBox((*theWindowObject).fMovie, &myMovieRect);
    // set the size and position of the overlay rectangle
    MacSetRect(&myPictRect, 0, 0, 32, 32);
    MacOffsetRect(&myPictRect,
        myMovieRect.right - (myPictRect.right + 5),
        myMovieRect.bottom - (myPictRect.bottom + 5));
    // draw the picture
    DrawPicture((*myAppData).fPicture, &myPictRect);
    return(noErr);
}
```

There's nothing very complicated in this prescreen buffer imaging completion procedure. Essentially, it just figures out where in the buffer to draw the picture and then draws it. We assume that a handle to the picture data is stored in the `fPicture` field of the application data record.

Intercepting QuickTime VR Manager Functions

Suppose we want to play a sound every time the user clicks on (that is, triggers) a hot spot. The easiest way to do this is to install an *intercept procedure* that is called each time a hot spot is triggered. The intercept procedure simply plays the sound and then returns, whereupon QuickTime VR processes the hot spot click as usual. **Listing 5** shows a simple hot spot triggering intercept procedure.

Listing 5: Playing a sound on hot spot clicks

```
MyInterceptRoutine
pascal void MyInterceptRoutine (
    QTVRInstance theInstance,
    QTVRInterceptPtr theMsg,
    WindowObject theWindowObject,
    Boolean *cancel)
```

```
{
#pragma unused(theInstance, theWindowObject)
    Boolean myCancelInterceptedProc = false;
    switch (theMsg->selector) {
        case kQTVRTriggerHotSpotSelector:
            MyPlaySound();
            break;
    }
    *cancel = myCancelInterceptedProc;
}
```

An intercept routine is executed whenever the intercepted routine is called, either programmatically or by a user action. On entry, the QuickTime VR Manager provides three pieces of information: the relevant QTVR instance, a pointer to an *intercept record*, and an application-defined reference constant, which we use here to pass in the window object. The intercept record (pointed to by the `theMsg` parameter) has this structure:

```
struct QTVRInterceptRecord {
    SInt32 reserved1;
    SInt32 selector;
    SInt32 reserved2;
    SInt32 reserved3;
    SInt32 paramCount;
    void *parameter[6];
}
```

For present purposes, we need to inspect only the selector field, which contains a value that indicates which intercepted routine is being called. As you can see in **Listing 5**, we look for any calls to `QTVRTriggerHotSpot` and call the application-defined function `MyPlaySound` when we get one.

We install an intercept procedure by calling the `QTVRInstallInterceptProc` function, as shown in **Listing 6**.

Listing 6: Installing an intercept routine

```
MyInstallInterceptRoutine
void MyInstallInterceptRoutine (
    QTVRInstance theInstance, WindowObject theWindowObject)
{
    QTVRInterceptUPP myInterceptProc;
    myInterceptProc =
        NewQTVRInterceptProc(MyInterceptRoutine);
    QTVRInstallInterceptProc(theInstance,
        kQTVRTriggerHotSpotSelector, myInterceptProc,
        (SInt32)theWindowObject, 0);
}
```

THE QUICKTIME VR FILE FORMAT

Unlike movies containing other interactive media types (such as sprite or Flash) where the media data can be stored in a single track, a QuickTime VR movie always contains several tracks. A panorama movie, for instance, contains a *panorama image track* (which holds the image data for the panorama), a *panorama track* (which contains information about the panoramic node), and a *QTVR track* (which maintains general information about the movie, such as the default imaging properties). Similarly, an object movie contains an *object image track* (which holds the image data for the object), an *object track* (which contains information about the object node), and a *QTVR track*. For multi-node movies, the QTVR track also contains a list of the nodes in the movie and an indication of which node is the default node. Movies with hot spots also contain a *hot spot image track* (a video track where the hot spots are designated by colored regions).

Usually this structure is important to us only when we want to create a QuickTime VR movie. But it's also useful when we want to

Backup

It's Not as Rough
as It Used to Be . . .

Automatic Backup System

*The Only Complete Software
and Hardware Solution!*

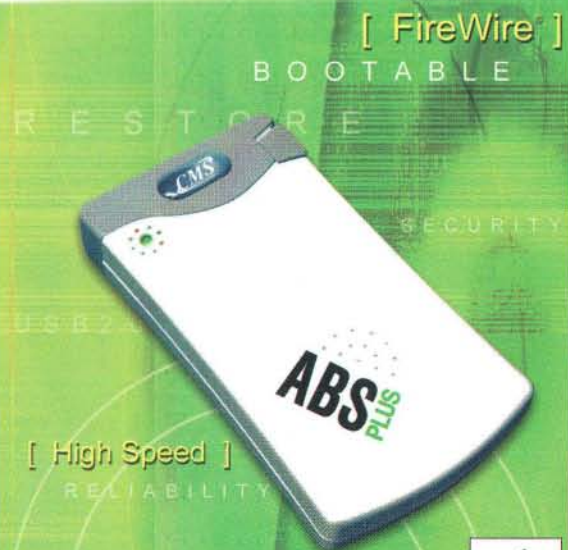
The ABS is The Easiest Way Invented
to Backup Your Mac®... Automatically!

Backup, store & share files between iMacs™, iBooks™,
Power Macs™, & PowerBooks™, including video,
digital photography, graphics, & MP3s up to 60GB!

For More Information Call **800 • 327 • 5773** or
click www.cmsproducts.com/mt1



© 2001 CMS Peripherals, Inc. and the CMS Logo are trademarks of CMS Peripherals, Inc. Mac and the Mac Logo are trademarks of Apple Computer, Inc. registered in the U.S. and other countries. All other trademarks are the property of their respective owners. All rights reserved.



**Now Get 15% Off
All Online Orders!**



alter an existing QuickTime VR movie or to extract information not provided by the available QuickTime VR Manager functions.

Working with Node Information

A QTVR track maintains general information about a QuickTime VR movie. Each individual sample in the QTVR track's media is an atom container called a *node information atom container*. This atom container holds a *node header atom*, which contains information about a single node, such as the node's type, ID, and name. The node information atom container can also hold a *hot spot parent atom* if the node has any hot spots in it. The QuickTime VR Manager provides the QTVRGetNodeInfo function that we can use to get a copy of a particular node information atom container or of any of its children. **Listing 7** defines a function that we can use to get a copy of a node header atom for a specified node ID.

Listing 7: Finding the node header atom data

```
QTVRUtils_GetNodeHeaderAtomData
OSErr QTVRUtils_GetNodeHeaderAtomData
    (QTVRInstance theInstance, UInt32 theNodeID,
     QTVRNodeHeaderAtomPtr theNodeHdrPtr)
{
    QAtomContainer    myNodeInfo;
    QAtom             myAtom;
    OSErr             myErr = noErr;
    // get the node information atom container for the specified node
    myErr = QTVRGetNodeInfo(theInstance, theNodeID,
        &myNodeInfo);
    if (myErr != noErr)
        return(myErr);
    // get the single node header atom in the node information atom container
    myAtom = QTFindChildByID(myNodeInfo,
        kParentAtomIsContainer, kQTVRNodeHeaderAtomType, 1,
        NULL);
    if (myAtom != 0)
        myErr = QTCopyAtomDataToPtr(myNodeInfo, myAtom, false,
            sizeof(QTVRNodeHeaderAtom), theNodeHdrPtr, NULL);
    else
        myErr = cannotFindAtomErr;
    QTDisposeAtomContainer(myNodeInfo);
    return(myErr);
}
```

As you can see, we call QTVRGetNodeInfo to get the node information atom container for the specified node ID; then we call QTFindChildByID to find the single node header atom inside that container. If we find that atom, we call QTCopyAtomDataToPtr to make a copy of its data. A node header atom has this structure:

```
struct QTVRNodeHeaderAtom {
    UInt16      majorVersion;
    UInt16      minorVersion;
    OSType      nodeType;
    QAtomID     nodeID;
    QAtomID     nameAtomID;
    QAtomID     commentAtomID;
    UInt32      reserved1;
    UInt32      reserved2;
};
```

Listing 8 defines the function QTVRUtils_GetNodeType, which reads the nodeType field of a node header atom to determine the node type. (In fact, the QuickTime VR Manager provides the QTVRGetNodeType function to get a node's type; we present QTVRUtils_GetNodeType simply to show another way of getting that information.)

Listing 8: Getting a node type

```
QTVRUtils_GetNodeType
OSErr QTVRUtils_GetNodeType (QTVRInstance theInstance,
    UInt32 theNodeID, OSType *theNodeType)
```

```
{
    QTVRNodeHeaderAtom myNodeHeader;
    OSErr myErr = noErr;
    // make sure we always return some meaningful value
    *theNodeType = kQTVRUnknownType;
    // get the node header atom data
    myErr = QTVRUtils_GetNodeHeaderAtomData(theInstance,
        theNodeID, &myNodeHeader);
    if (myErr == noErr)
        *theNodeType = EndianU32_BtoN(myNodeHeader.nodeType);
    return(myErr);
}
```

There is no need to deallocate the block of data returned by QTVRUtils_GetNodeHeaderAtomData because it is allocated on the stack in a local variable.

Working with a VR World

All samples in a QTVR track use a single sample description. The data field of that sample description holds a *VR world atom container*, which holds general information about the scene contained in the QuickTime VR movie, including the name of the entire scene, the default node ID, and the default imaging properties. We can use the QTVRGetVRWorld function to retrieve a copy of a movie's VR world atom container. **Listing 9** illustrates how to use this function.

Listing 9: Finding the VR world atom data

```
QTVRUtils_GetVRWorldHeaderAtomData
OSErr QTVRUtils_GetVRWorldHeaderAtomData
    (QTVRInstance theInstance,
     QTVRWorldHeaderAtomPtr theVRWorldHdrAtomPtr)
{
    QAtomContainer    myVRWorld;
    QAtom             myAtom;
    OSErr             myErr = noErr;
    // get the VR world
    myErr = QTVRGetVRWorld(theInstance, &myVRWorld);
    if (myErr != noErr)
        return(myErr);
    // get the single VR world header atom in the VR world
    myAtom = QTFindChildByIndex(myVRWorld,
        kParentAtomIsContainer, kQTVRWorldHeaderAtomType, 1,
        NULL);
    if (myAtom != 0)
        myErr = QTCopyAtomDataToPtr(myVRWorld, myAtom, false,
            sizeof(QTVRWorldHeaderAtom), theVRWorldHdrAtomPtr,
            NULL);
    else
        myErr = cannotFindAtomErr;
    QTDisposeAtomContainer(myVRWorld);
    return(myErr);
}
```

A VR world atom container contains (perhaps among other things) a single *VR world header atom*, whose structure is defined by the QTVRWorldHeaderAtom data type:

```
struct QTVRWorldHeaderAtom {
    UInt16      majorVersion;
    UInt16      minorVersion;
    QAtomID     nameAtomID;
    UInt32      defaultNodeID;
    UInt32      vrWorldFlags;
    UInt32      reserved1;
    UInt32      reserved2;
};
```

We can use this information to determine the node ID of a scene's default node, as shown in **Listing 10**.

Listing 10: Finding a scene's default node

```
QTVRUtils_GetDefaultNodeID
UInt32 QTVRUtils_GetDefaultNodeID (QTVRInstance theInstance)
{
```

```

QTVRWorldHeaderAtom    myVRWorldHeader;
UInt32                 myNodeID = kQTVRCurrentNode;
OSErr                 myErr = noErr;
myErr = QTVRUtils_GetVRWorldHeaderAtomData(theInstance,
    &myVRWorldHeader);
if (myErr == noErr)
    myNodeID = EndianU32_BtoN(myVRWorldHeader.defaultNodeID);
return(myNodeID);
}

```

QTVRUtils_GetDefaultNodeID can be useful if we need to know the ID of a movie's default node, since there is no QuickTime VR Manager function that returns this information directly.

WIRED ACTIONS AND QUICKTIME VR

In a handful of recent articles, we've seen how to work with QuickTime wired actions in conjunction with sprite tracks, text tracks, and Flash tracks. We use wired actions to attach dynamic, interactive behaviors to elements in a QuickTime movie and to allow different elements in those movies (and indeed in different movies) to communicate with one another. In this section, we'll investigate how to work with wired actions and QuickTime VR movies.

Sending Actions to QuickTime VR Movies

Let's begin by taking a look at the wired actions that can be targeted at a QuickTime VR movie. When action wiring was first introduced, in QuickTime 3, these five wired actions were supported:

```

enum {
    kActionQTVRSetPanAngle      = 4096,
    kActionQTVRSetTiltAngle     = 4097,
    kActionQTVRSetFieldOfView   = 4098,
    kActionQTVRShowDefaultView  = 4099,
    kActionQTVRGoToNodeID       = 4100
};

```

The first three actions allow us to set a new pan angle, tilt angle, or field of view in a QuickTime VR movie. Each of these actions takes a single parameter, a value of type float that specifies the desired new angle. This value should be specified in degrees (not radians) and is by default an absolute angle to pan, tilt, or zoom to. It's often useful to specify a relative value instead; we can indicate that the parameter value is relative by inserting into the action atom an atom of type kActionFlags whose atom data is a long integer with (at least) the kActionFlagActionIsDelta flag set. **Listing 11** shows how we can build an atom container holding a wired atom that pans the target QuickTime VR movie one degree to the left each time it gets an idle event. (We'll see later how to make sure the movie is sent idle events.)

Listing 11: Panning a QuickTime VR movie during idle events

```

AddVRAct_CreateIdleActionContainer
static OSERR AddVRAct_CreateIdleActionContainer
    (QTAtomContainer *theActions)
{
    QTAtom    myEventAtom = 0;
    QTAtom    myActionAtom = 0;
    long      myAction;
    float      myPanAngle = 1.0;
    UInt32     myFlags;
    OSERR      myErr = noErr;
    myErr = QTNewAtomContainer(theActions);
    if (myErr != noErr)
        goto bail;
    myErr = QTInsertChild(*theActions, kParentAtomIsContainer,

```

```

        kQTEventIdle, 1, 1, 0, NULL, &myEventAtom);
    if (myErr != noErr)
        goto bail;
    myErr = QTInsertChild(*theActions, myEventAtom, kAction,
        1, 1, 0, NULL, &myActionAtom);
    if (myErr != noErr)
        goto bail;
    myAction = EndianS32_NtoB(kActionQTVRSetPanAngle);
    myErr = QTInsertChild(*theActions, myActionAtom,
        kWhichAction, 1, 1, sizeof(long), &myAction, NULL);
    if (myErr != noErr)
        goto bail;
    AddVRAct_ConvertFloatToBigEndian(&myPanAngle);
    myErr = QTInsertChild(*theActions, myActionAtom,
        kActionParameter, 1, 1, sizeof(float), &myPanAngle,
        NULL);
    if (myErr != noErr)
        goto bail;
    myFlags = EndianU32_NtoB(kActionFlagActionIsDelta |
        kActionFlagParameterWrapsAround);
    myErr = QTInsertChild(*theActions, myActionAtom,
        kActionFlags, 1, 1, sizeof(UInt32), &myFlags, NULL);
bail:
    return(myErr);
}

```

The action kActionQTVRShowDefaultView sets the current node to its default view (that is, the view that is displayed when the node is first entered). The kActionQTVRGoToNodeID action takes a single parameter that specifies a node ID; when the action is executed, the node with that ID becomes the current node. QuickTime 3 also introduced four wired action operands, which we can use to get information about the current state of a QuickTime VR movie:

```

enum {
    kOperandQTVRPanAngle        = 4096,
    kOperandQTVRTiltAngle       = 4097,
    kOperandQTVRFieldOfView     = 4098,
    kOperandQTVRNodeID          = 4099
};

```

QuickTime 5 added three more actions that we can send to a QuickTime VR movie:

```

enum {
    kActionQTVREnableHotSpot     = 4101,
    kActionQTVRShowHotSpots      = 4102,
    kActionQTVRTranslateObject   = 4103
};

```

The kActionQTVREnableHotSpot action enables or disables a hot spot. This action requires two parameters, a long integer that specifies a hot spot ID and a Boolean value that specifies whether to enable (true) or disable (false) the hot spot. The kActionQTVRShowHotSpots action shows or hides all hot spots in a node, depending on the Boolean value in the parameter atom. The kActionQTVRTranslateObject action sets the view center of an object node to the values specified in the action's two parameters. To allow us to retrieve the current hot spot visibility state and the current view center, QuickTime 5 introduced three additional operands:

```

enum {
    kOperandQTVRHotSpotsVisible = 4100,
    kOperandQTVRViewCenterH     = 4101,
    kOperandQTVRViewCenterV     = 4102
};

```

There is currently no operand that will allow us to determine whether a particular hot spot is enabled.

Adding Actions to QuickTime VR Movies

We can add two kinds of wired actions to QuickTime VR movies: (1) actions that are associated with a particular node and (2) actions that are associated with a particular hot spot in a node. Examples of node-specific actions are setting the pan and tilt angles when the user first enters the node and performing some actions periodically when the movie gets an idle event. An example of a hot-spot-specific action might be playing a sound when the cursor is moved over a hot spot.

All QuickTime VR wired actions are attached to a particular node, so the atom containers holding the actions are placed in the node information atom container that is contained in the media sample for that node in the QTVR track. So, our job here boils down to finding a media sample in the QTVR track, constructing some atom containers for our desired actions, placing those action containers into the appropriate places in the media sample, and then writing the modified media sample back into the QTVR track. We'll also need to put an atom into the media property atom container of the QTVR track to enable wired action and idle event processing.

Adding Actions to a Hot Spot

Let's begin by seeing how to attach some wired actions to a particular hot spot in a node. Let's suppose that we know both the node ID and the hot spot ID, and that we have already constructed the atom container that holds the wired actions. Recall that a QTVR track contains one media sample for each node in the movie and that that media sample is a node information atom container. For simplicity, we'll assume that we want to wire a hot spot in a single-node QuickTime VR movie. As a result, we can get the media sample by calling `GetMediaSample`, like this:

```
GetMediaSample(myMedia, mySample, 0, NULL, myMediaTime, NULL,
    &mySampleDuration, (SampleDescriptionHandle)myQTVRDesc,
    NULL, 1, NULL, &mySampleFlags);
```

If `GetMediaSample` returns successfully, then `mySample` will be the atom container that holds the atoms we want to modify.

At this point, we'll call an application function `AddVRAct_SetWiredActionsToHotSpot` to add our wired actions to the specified hot spot:

```
AddVRAct_SetWiredActionsToHotSpot(mySample, myHotSpotID,
    myActions);
```

The first thing we need to do in `AddVRAct_SetWiredActionsToHotSpot` is find the hot spot parent atom inside the node information atom container:

```
myHotSpotParentAtom = QTFindChildByIndex(theSample,
    kParentAtomIsContainer, kQTVRHotSpotParentAtomType,
    1, NULL);
```

A hot spot parent atom contains a *hot spot atom* (of type `kQTVRHotSpotAtomType`) for each hot spot in the node. The ID of the hot spot atom is the same as the ID of the hot spot, so we can find the appropriate hot spot atom like this:

```
myHotSpotAtom = QTFindChildByID(theSample,
    myHotSpotParentAtom, kQTVRHotSpotAtomType,
    theHotSpotID, NULL);
```

We add wired actions to a hot spot by inserting an event atom

(that is, an atom of type `kQTEventType`) into the hot spot atom:

```
QTInsertChildren(theSample, myHotSpotAtom, theActions);
```

Listing 12 shows our complete definition of `AddVRAct_SetWiredActionsToHotSpot`.

Listing 12: Adding wired actions to a hot spot

```
static OSERR AddVRAct_SetWiredActionsToHotSpot
    (Handle theSample, long theHotSpotID,
    QTAAtomContainer theActions)
{
    QTAAtom    myHotSpotParentAtom = 0;
    QTAAtom    myHotSpotAtom = 0;
    short      myCount, myIndex;
    OSERR      myErr = paramErr;
    myHotSpotParentAtom = QTFindChildByIndex(theSample,
        kParentAtomIsContainer, kQTVRHotSpotParentAtomType,
        1, NULL);
    if (myHotSpotParentAtom == NULL)
        goto bail;
    myHotSpotAtom = QTFindChildByID(theSample,
        myHotSpotParentAtom, kQTVRHotSpotAtomType,
        theHotSpotID, NULL);
    if (myHotSpotAtom == NULL)
        goto bail;
    // see how many events are already associated with the specified hot spot
    myCount = QTCountChildrenOfType(theSample, myHotSpotAtom,
        kQTEventType);
    for (myIndex = myCount; myIndex > 0; myIndex--) {
        QTAAtom    myTargetAtom = 0;
        // remove all the existing events
        myTargetAtom = QTFindChildByIndex(theSample,
            myHotSpotAtom, kQTEventType, myIndex, NULL);
        if (myTargetAtom != 0) {
            myErr = QTRemoveAtom(theSample, myTargetAtom);
            if (myErr != noErr)
                goto bail;
        }
    }
    if (theActions) {
        myErr = QTInsertChildren(theSample, myHotSpotAtom,
            theActions);
        if (myErr != noErr)
            goto bail;
    }
    bail:
    return(myErr);
}
```

You'll notice that we look to see whether the hot spot atom already contains any event atoms; if so, we remove them from the hot spot atom. This ensures that the event atom we pass to `AddVRAct_SetWiredActionsToHotSpot` is the only one in the hot spot atom.

Adding Actions to a Node

We add wired actions to a node by inserting children into the node information atom container for that node. The type of a child atom for a wired action should be the same as the event type, and the ID should be 1. **Listing 13** defines the `AddVRAct_SetWiredActionsToNode` function, which we use to add a wired atom to a particular node. The first parameter is assumed to be the node information atom container.

Listing 13: Adding wired actions to a node

```
static OSERR AddVRAct_SetWiredActionsToNode
    (Handle theSample, QTAAtomContainer theActions,
    UInt32 theActionType)
{
    QTAAtom    myEventAtom = 0;
    QTAAtom    myTargetAtom = 0;
    OSERR      myErr = noErr;
    // look for an event atom in the specified actions atom container
```

Peachpit Press and Macromedia Press

offer the best selection for designers and developers

Save 15%
off these
titles at

<http://www.peachpit.com/forms/flashMX.asp>



**Macromedia Flash MX
Advanced for Windows
and Macintosh: Visual
QuickPro Guide**
Russell Chun 0-
201-75846-6 • \$29.99



**Macromedia Flash MX
for Windows and
Macintosh: Visual
QuickStart Guide**
Katherine Ulrich
0-201-79481-0 • \$21.99



**Macromedia Flash MX:
Training from the Source**
Chrissy Rey
0-201-79482-9 • \$44.99



**Macromedia Flash MX
Game Design Demystified**
Jobe Makar
0-201-77021-0 • \$49.99
*Coming soon



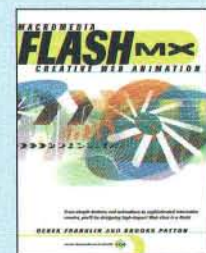
**Macromedia Flash MX
ActionScripting:
Advanced Training
from the Source**
Derek Franklin &
Jobe Makar
0-201-77022-9 • \$44.99



**Reality ColdFusion:
Flash MX Integration**
Ben Forta, et al.
0-321-12515-0 • \$39.99
*Coming soon



**Macromedia Flash MX
FreeHand 10: Advanced
Training from the Source**
Brad Kozak
0-201-77502-6 • \$44.99



**Macromedia Flash MX
Creative Web Animation**
Derek Franklin
0-321-11785-9 • \$39.99
*Coming soon



```

if (theActions != NULL)
    myEventAtom = QTFindChildByID(theActions,
        kParentAtomIsContainer, theActionType, 1, NULL);
// look for an event atom in the node information atom container
myTargetAtom = QTFindChildByID(theSample,
    kParentAtomIsContainer, theActionType, 1, NULL);
if (myTargetAtom != 0) {
    // if there is already an event atom in the node information atom container,
    // then either replace it with the one we were passed or remove it
    if (theActions != NULL)
        myErr = QTReplaceAtom(theSample, myTargetAtom,
            theActions, myEventAtom);
    else
        myErr = QTRemoveAtom(theSample, myTargetAtom);
} else {
    // there is no event atom in the node information atom container,
    // so add in the one we were passed
    if (theActions != NULL)
        myErr = QTInsertChildren(theSample,
            kParentAtomIsContainer, theActions);
}
return(myErr);
}

```

We can add an idle event handler to a node like this:

```

AddVRAct_SetWiredActionsToNode(mySample, myActions,
    kQTEventIdle);

```

And we can add a frame-loaded event handler to a node like this:

```

AddVRAct_SetWiredActionsToNode(mySample, myActions,
    kQTEventFrameLoaded);

```

Other event types (such as `kQTEventMouseClicked` or `kQTEventKey`) might not make sense for a node-based wired action.

Updating the Media Property Atom

When we added some wiring to a sprite track, we needed to include in the track's media property atom an atom of type `kSpriteTrackPropertyHasActions` whose atom data is set to true. (See "Wired", in *MacTech*, May 2001.) This atom tells the movie controller that the sprite track has wiring associated with it. If, in addition, any of the wired sprites employs the `kQTEventIdle` event, we also need to add an atom of type `kSpriteTrackPropertyQTIdleEventsFrequency` whose atom data indicates the desired idle event frequency, in ticks. We need to add these same atoms to the media property atom when we wire a QuickTime VR movie. **Listing 14** defines the function `AddVRAct_WriteMediaPropertyAtom`, which we use to add the appropriate atoms.

Listing 14: Adding atoms to the media property atom

```

AddVRAct_WriteMediaPropertyAtom
static OSERr AddVRAct_WriteMediaPropertyAtom (Media theMedia,
    long thePropertyID, long thePropertySize,
    void *theProperty)
{
    QTAtomContainer    myPropertyAtom = NULL;
    QTAtom             myAtom = 0;
    OSERr              myErr = noErr;
    // get the current media property atom
    myErr = GetMediaPropertyAtom(theMedia, &myPropertyAtom);
    if (myErr != noErr)
        goto bail;
    // if there isn't one yet, then create one
    if (myPropertyAtom == NULL) {
        myErr = QTNewAtomContainer(&myPropertyAtom);
        if (myErr != noErr)
            goto bail;
    }
    // see if there is an existing atom of the specified type; if not, then create one
    myAtom = QTFindChildByID(myPropertyAtom,
        kParentAtomIsContainer, thePropertyID, 1, NULL);
    if (myAtom == NULL) {
        myErr = QTInsertChild(myPropertyAtom,

```

```

        kParentAtomIsContainer, thePropertyID, 1, 0, 0, NULL,
        &myAtom);
    if ((myErr != noErr) || (myAtom == NULL))
        goto bail;
    // set the data of the specified atom to the data passed in
    myErr = QTSetAtomData(myPropertyAtom, myAtom,
        thePropertySize, (Ptr)theProperty);
    if (myErr != noErr)
        goto bail;
    // write the new atom data out to the media property atom
    myErr = SetMediaPropertyAtom(theMedia, myPropertyAtom);
bail:
    if (myPropertyAtom != NULL)
        myErr = QTDisposeAtomContainer(myPropertyAtom);
    return(myErr);
}

```

To indicate that the QuickTime VR movie has wired actions embedded in it, we can call `AddVRAct_WriteMediaPropertyAtom` like this:

```

myHasActions = true;
AddVRAct_WriteMediaPropertyAtom(myMedia,
    kSpriteTrackPropertyHasActions,
    sizeof(Boolean), &myHasActions);

```

And we can set the idle frequency like this:

```

myFrequency = EndianU32_NtoB(30);
AddVRAct_WriteMediaPropertyAtom(myMedia,
    kSpriteTrackPropertyQTIdleEventsFrequency,
    sizeof(UInt32), &myFrequency);

```

Saving the Modified Media Data

So far, we've added some wired atoms to a node information atom container or to a hot spot atom inside of a node information atom container, and we've updated the media property atom of the QTVR track. To save these changes, we need to replace the appropriate sample in the QTVR track media and then update the movie atom. **Listing 15** shows the complete definition of the `AddVRAct_AddWiredActionsToQTVRMovie` function, which we use to wire a QuickTime VR movie.

Listing 15: Adding wired actions to a QuickTime VR movie

```

AddVRAct_AddWiredActionsToQTVRMovie
static void AddVRAct_AddWiredActionsToQTVRMovie
    (FSSpec *theFSSpec)
{
    short              myResID = 0;
    short              myResRefNum = -1;
    Movie              myMovie = NULL;
    Track              myTrack = NULL;
    Media              myMedia = NULL;
    TimeValue          myTrackOffset;
    TimeValue          myMediaTime;
    TimeValue          mySampleDuration;
    TimeValue          mySelectionDuration;
    TimeValue          myNewMediaTime;
    QTVRSampleDescriptionHandle
        myQTVRDesc = NULL;
    Handle              mySample = NULL;
    short              mySampleFlags;
    Fixed              myTrackEditRate;
    QTAtomContainer    myActions = NULL;
    Boolean             myHasActions;
    long               myHotSpotID = 0L;
    UInt32              myFrequency;
    OSERr              myErr = noErr;
    // open the movie file and get the QTVR track from the movie
    // open the movie file for reading and writing
    myErr = OpenMovieFile(theFSSpec, &myResRefNum, fsRdWrPerm);
    if (myErr != noErr) goto bail; myErr =
        NewMovieFromFile(&myMovie, myResRefNum, &myResID,
            NULL, newMovieActive, NULL);
    if (myErr != noErr)
        goto bail;
}

```

```

// find the first QTVR track in the movie;
myTrack = GetMovieIndTrackType(myMovie, 1, kQTVRQTVRType,
    movieTrackMediaType);
if (myTrack == NULL)
    goto bail;
// get the first media sample in the QTVR track
myMedia = GetTrackMedia(myTrack);
if (myMedia == NULL)
    goto bail;
myTrackOffset = GetTrackOffset(myTrack);
myMediaTime = TrackTimeToMediaTime(myTrackOffset, myTrack);
// allocate some storage to hold the sample description for the QTVR track
myQTVRDesc = (QTVRSampleDescriptionHandle)NewHandle(4);
if (myQTVRDesc == NULL)
    goto bail;
mySample = NewHandle(0);
if (mySample == NULL)
    goto bail;
myErr = GetMediaSample(myMedia, mySample, 0, NULL,
    myMediaTime, NULL, &mySampleDuration,
    (SampleDescriptionHandle)myQTVRDesc, NULL, 1, NULL,
    &mySampleFlags);
if (myErr != noErr)
    goto bail;
// add idle actions
// create an action container for idle actions
myErr = AddVRAct_CreateIdleActionContainer(&myActions);
if (myErr != noErr)
    goto bail;
// add idle actions to sample
myErr = AddVRAct_SetWiredActionsToNode(mySample, myActions,
    kQTEventIdle);
if (myErr != noErr)
    goto bail;
myErr = QTDisposeAtomContainer(myActions);
if (myErr != noErr)
    goto bail;
// add frame-loaded actions
// create an action container for frame-loaded actions
myErr = AddVRAct_CreateFrameLoadedActionContainer
    (&myActions);
if (myErr != noErr)
    goto bail;
// add frame-loaded actions to sample
myErr = AddVRAct_SetWiredActionsToNode(mySample, myActions,
    kQTEventFrameLoaded);
if (myErr != noErr)
    goto bail;
myErr = QTDisposeAtomContainer(myActions);
if (myErr != noErr)
    goto bail;
// add hot-spot actions
// find the first hot spot in the selected node; don't bail if there are no hot spots
myErr = AddVRAct_GetFirstHotSpot(mySample, &myHotSpotID);
if ((myErr == noErr) && (myHotSpotID != 0)) {
    // create an action container for hot-spot actions
    myErr = AddVRAct_CreateHotSpotActionContainer
        (&myActions);
    if (myErr != noErr)
        goto bail;
    // add hot-spot actions to sample
    myErr = AddVRAct_SetWiredActionsToHotSpot(mySample,
        myHotSpotID, myActions);
    if (myErr != noErr)
        goto bail;
}
// replace sample in media
myTrackEditRate = GetTrackEditRate(myTrack, myTrackOffset);
if (GetMoviesError() != noErr)
    goto bail;
GetTrackNextInterestingTime(myTrack, nextTimeMediaSample |
    nextTimeEdgeOK, myTrackOffset, fixed1, NULL,
    &mySelectionDuration);
if (GetMoviesError() != noErr)
    goto bail;
myErr = DeleteTrackSegment(myTrack, myTrackOffset,
    mySelectionDuration);
if (myErr != noErr)
    goto bail;
myErr = BeginMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;
myErr = AddMediaSample( myMedia,
    mySample,
    0,
    GetHandleSize(mySample),

```

```

    mySampleDuration,
    (SampleDescriptionHandle)myQTVRDesc,
    1,
    mySampleFlags,
    &myNewMediaTime);
if (myErr != noErr)
    goto bail;
myErr = EndMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;
// add the media to the track
myErr = InsertMediaIntoTrack(myTrack, myTrackOffset,
    myNewMediaTime, mySelectionDuration,
    myTrackEditRate);
if (myErr != noErr)
    goto bail;
// set the media property atom to enable wired action and idle-time processing
myHasActions = true;
myErr = AddVRAct_WriteMediaPropertyAtom(myMedia,
    kSpriteTrackPropertyHasActions, sizeof(Boolean),
    &myHasActions);
if (myErr != noErr)
    goto bail;
myFrequency = EndianU32_NtoB(1);
myErr = AddVRAct_WriteMediaPropertyAtom(myMedia,
    kSpriteTrackPropertyQTIdleEventsFrequency,
    sizeof(UInt32), &myFrequency);
if (myErr != noErr)
    goto bail;
// update the movie resource
myErr = UpdateMovieResource(myMovie, myResRefNum, myResID,
    NULL);
if (myErr != noErr)
    goto bail;
// close the movie file
myErr = CloseMovieFile(myResRefNum);
bail:
if (myActions != NULL)
    QTDisposeAtomContainer(myActions);
if (mySample != NULL)
    DisposeHandle(mySample);
if (myQTVRDesc != NULL)
    DisposeHandle((Handle)myQTVRDesc);
if (myMovie != NULL)
    DisposeMovie(myMovie);
}

```

CONCLUSION

In this article, we've learned how to work with the QuickTime VR Manager to control the operation of QuickTime VR movies programmatically. We've seen how to adjust pan, tilt, and zoom angles, how to alter the displayed image by drawing into a panorama's prescreen buffer, and how to intercept some QuickTime VR Manager functions. As usual, these few examples of using the VR APIs are just the tip of the iceberg; with just a little bit more time and energy, we can develop some even more impressive interactive applications using QuickTime VR.

We've also taken a look at QuickTime VR and wired actions, first reviewing how to send actions to VR movies and then (more importantly) learning how to embed wired actions into QuickTime VR movies. We haven't yet learned how to actually create QuickTime VR movies from scratch, but we do have a preliminary idea of how they are put together (at least in part). Perhaps in a future article we'll learn how to build QuickTime VR movies.

CREDITS AND REFERENCES

Thanks to Bryce Wolfson for reviewing an earlier version of this article and for providing some helpful comments. The code for adding wired actions to QuickTime VR movies is based on some code by Bill Wright. For complete information on the QuickTime VR Manager, see the book *Virtual Reality Programming With QuickTime VR 2.1* by Apple Computer, Inc.

List of Advertisers

ab DataTools	74
AEC Software	59
Aladdin Knowledge Systems, Inc.	11
Aladdin Systems, Inc.	67
Appgen Business Software, Inc.	57
Ascending Technologies	37
Avesta Computer Services, Ltd.	26
Big Nerd Ranch, Inc.	60
Blue World Communications, Inc.	17
Borland Software Corporation	15
CMS Peripherals Inc	81
Computer Systems Odessa, Corp.	23
Crescendo Software	75
DevDepot	28, 29
Dr. Bott LLC	79
Einhugur Programming Resources	75
Electric Butterfly	74
Exabyte	25
FairCom Corporation	1
Felt Tip Software	19
Fetch Softworks	5
Flickinger Software	75
Full Spectrum Software, Inc.	31
IDG World Expo Corporation	48
James Sentman Software	76
Jiiva, Inc.	7
Kaidan	21
MacDirectory	41
MacTech Central	53
MacTech Magazine	39
Management Software Inc.	47
Mathemaesthetics, Inc.	51
Metrowerks Corporation	BC
MYOB US, Inc.	69
Netopia, Inc.	45
O'Reilly & Associates, Inc.	61
OpenBase International, Ltd.	71
OpenLink Software Inc.	13
Paradigma Software	56
Peachpit Press	85
Perforce Software, Inc.	89
piDog Software	74
PrimeBase (SNAP Innovation)	9
REAL Software, Inc.	35
RiverSong InterActive	76
Runtime Revolution Limited	IFC
Scientific Placement, Inc.	19
Small Dog Electronics	33
Stone Design Corp	55
Sustainable Softworks	50
Thursby Software Systems, Inc.	65
WIBU-SYSTEMS AG	43
Xochi Media Inc.	76
Zero G Software	63

List of Products

Accessories • DevDepot	28
Accessories • Dr. Bott LLC	79
Accounting Products • Appgen Business Software, Inc.	57
Adobe Press • Peachpit Press	85
Application Builder Collection • Jiiva, Inc.	7
Backup Systems & Peripherals • CMS Peripherals Inc	81
Big Nerd Ranch • Big Nerd Ranch, Inc.	60
Books • O'Reilly & Associates, Inc.	61
c-tree Plus • FairCom Corporation	1
CalendarMonster • Ascending Technologies	37
Career Opportunities • Scientific Placement, Inc.	19
CodeWarrior • Metrowerks Corporation	90
Concept Draw • Computer Systems Odessa, Corp.	23
Consulting & Training Services • Avesta Computer Services, Ltd.	26
Corona • Flickinger Software	75
DAVE • Thursby Software Systems, Inc.	65
db Reports • ab DataTools	74
Development & Testing • Full Spectrum Software, Inc.	31
FastTrack • AEC Software	59
Fetch • Fetch Softworks	5
InstallAnywhere • Zero G Software	63
InstallerMaker, StuffIt • Aladdin Systems, Inc.	67
IPNetRouter & IPNetSentry • Sustainable Softworks	50
iScreensaver Designer • Xochi Media Inc.	76
JBuilder • Borland Software Corporation	15
JobOrder • Management Software Inc.	47
Lasso • Blue World Communications, Inc.	17
MacDirectory • MacDirectory	41
MacTech Central • MacTech Central	53
MacTech Magazine Subscription • MacTech Magazine	39
Macworld Conference & Expo • IDG World Expo Corporation	48
MYOB • MYOB US, Inc.	69
OpenBase • OpenBase International, Ltd.	71
Photographic VR Solutions • Kaidan	21
Picture Play • Crescendo Software	75
piDog Utilities • piDog Software	74
PrimeBase • PrimeBase (SNAP Innovation)	9
REALbasic Plug-ins • Einhugur Programming Resources	75
REALbasic • REAL Software, Inc.	35
Resorcerer • Mathemaesthetics, Inc.	51
Revolution • Runtime Revolution Limited	ifc2
SCM Software • Perforce Software, Inc.	89
SmallDog.com • Small Dog Electronics	33
Software and Hardware • Aladdin Knowledge Systems, Inc.	11
Software Protection • WIBU-SYSTEMS AG	43
Sound Studio • Felt Tip Software	19
Stone Studio • Stone Design Corp	55
Timbuktu Pro & netOctopus • Netopia, Inc.	45
TitleTrack Jukebox • RiverSong InterActive	76
UniHelp Module • Electric Butterfly	74
Universal Data Access • OpenLink Software Inc.	13
Valentina • Paradigma Software	56
VXA • Exabyte	25
Whistle Blower • James Sentman Software	76

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

To meet deadlines, developers have two choices:

1. Use Perforce
2. Cut Corners



For developers under pressure to manage source code and do more in less time, Perforce's Fast Software Configuration Management System is the must-have tool.

With rival SCM systems, the only way to quicken the pace is to cut corners - but in the long run you pay the price with missed deadlines, uncertain contents, buggy releases and no way back to previous builds.

With Perforce, the fast way is always the right way. Install it fast, learn it fast, execute operations fast. With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean fast is right. See how Perforce compares with other leading SCM systems at <http://www.perforce.com/perforce/reviews.html>

Run at full speed even with hundreds of users and millions of files. At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

Work anywhere. Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronizing client workspace with server repository contents.

Develop and maintain multiple codelines.

Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

Truly cross platform. Perforce runs on more than 50 operating systems, including Windows and nearly every UNIX variation, from Linux and Mac OS X to AS400 and more.

Integrate with leading IDEs and defect trackers:

Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

Fast Software Configuration Management www.perforce.com

Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com
Free (and friendly) technical support is on hand to answer any and all evaluation questions.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.



CodeWarrior Rocks

CodeWarrior™ development tools are the #1 choice for professional Macintosh® developers. Now get ready for the encore. With the new CodeWarrior tools for Mac OS v8, you can build applications with both Cocoa® and the PowerPlant™ C++ framework, all within one integrated development environment. And you can write code quicker than ever before, because code-completion for C, C++, and Java™ gives you fast access to prototypes and parameters. Choose CodeWarrior tools – and get ready for a great performance.

To get more information or to request a white paper, email dskmrkt@metrowerks.com

CodeWarrior
Development Tools for Mac® OS

© Copyright. 2002. Metrowerks Corp. CodeWarrior, PowerPlant, Metrowerks and the Metrowerks logo are trademarks or registered trademarks of Metrowerks Corp. in the U.S. and/or other countries. Macintosh, Mac and Cocoa are trademarks and/or registered trademarks of Apple Computer, Inc. Java is a trademark of Sun Microsystems. ALL RIGHTS RESERVED. M690845A-MTC0602

